Oracle® Cloud

Using Oracle Autonomous Database Serverless

F61232-45
May 2024

Oracle Cloud Using Oracle Autonomous Database Serverless,

F61232-45

Contents

Preface

1 Overview

2 Quick Start

3 Tasks

4 Tutorials

5 Features

6 Reference

Preface

This document describes how to manage, monitor, and use Oracle Autonomous Database and provides references to related documentation.

Audience

This document is intended for Oracle Cloud users who want to manage and monitor Oracle Autonomous Database.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

Depending on the region and when you provisioned your database, and in some cases depending on your provisioning choice, the Oracle Database version for your Autonomous Database is one of: Oracle Database 19c, Oracle Database 21c, or Oracle Database 23ai.

If you have Oracle Database 19c, then many database concepts and features of this service are further documented here:

Oracle Database 19c

Oracle Database 21c

If you are using Always Free Autonomous Database with Oracle Database 21c, then many concepts and features of this service are further documented here:

Oracle Database 21c

Oracle Database 23ai

If you are using Always Free Autonomous Database with Oracle Database 23ai, then many concepts and features of this service are further documented here:

Oracle Database 23ai

For additional information, see these Oracle resources:

- Welcome to Oracle Cloud Infrastructure
- *Oracle Cloud Infrastructure Object Storage*
- Get Started Using Autonomous JSON Database
- GoldenGate Real-Time Data Replication in Cloud
- *Getting Started with Oracle Analytics Cloud*
- *User's Guide for Oracle Analytics Desktop*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Overview

- What's New for Oracle Autonomous Database Serverless
  Here's a summary of the noteworthy Oracle Autonomous Database additions and enhancements.
- About Autonomous Database
  Provides information about Autonomous Database features, workload types, and regions.
- Get Help, Search Forums, and Contact Support
  When you use Oracle Autonomous Database, sometimes you need to get help from the community or to talk to someone in Oracle support. This topic provides more information about getting help by viewing and posting questions on forums and using Oracle Cloud Support to create a support request.
- Service Level Objectives (SLOs)
  Describes the Service Level Objectives (SLOs) for Oracle Autonomous Database Serverless.
- How Is Autonomous Database Billed?
  Provides details for Autonomous Database billing.
- Always Free Autonomous Database
  You have the option to create a limited number of Always Free Autonomous Databases that do not consume cloud credits. Always Free databases can be created in Oracle Cloud Infrastructure accounts that are in a trial period, have paying status, or are always free. This section describes configuration differences, restrictions, and additional details for Always Free databases.

What's New for Oracle Autonomous Database Serverless

Here's a summary of the noteworthy Oracle Autonomous Database additions and enhancements.

See Previous Feature Announcements for 2023 announcements and older announcements.

- May 2024
- April 2024
- March 2024
- February 2024
- January 2024

May 2024

Feature	Description
Allow Different ACLs on Cross-Region Disaster Recovery Peer	You can independently modify network ACLs on a remote disaster recovery peer database. See Update Remote Peer Network ACLs and Update Remote Peer Network ACLs for a Backup-Based Disaster Recovery Peer for more information.
Documentation Addition: Best Practices for Low Latency Connections	Provides information on steps you can take to reduce the latency for connections between an application and Autonomous Database. See Best Practices for Low Latency Connections with Autonomous Database for more information.
Always Free with Oracle Database 23ai	The Oracle Database versions available for Always Free Autonomous Database are: Oracle Database 19c or Oracle Database 23ai. See Always Free Autonomous Database for more information.
Oracle Autonomous Database Free Container Image with Oracle Database 23ai	The Oracle Database versions available for Oracle Autonomous Database Free Container Image are: Oracle Database 19c or Oracle Database 23ai. See Use the Oracle Autonomous Database Free Container Image for more information.

April 2024

Feature	Description
Cross-Region Cloud Links	You can use Cloud Links in multiple regions, where the source region contains the data set's source database and one or more remote regions contain refreshable clones of the source database. See Register or Unregister a Data Set in a Different Region for more information.
Test Workloads Against an Upcoming Patch	Using the workload auto replay feature you can automatically capture a workload from a production database that is on the regular patch level and replay the workload on a target refreshable clone that is on the early patch level. This feature allows you to test an upcoming patch by running an existing workload that is in production against a patch before the patch reaches production. See Test Workloads Against an Upcoming Patch for more information.
Cloning Permissions	You can control and limit cloning operations on Autonomous Database with IAM permissions. See Cloning Permissions in IAM Permissions and API Operations for Autonomous Database for more information.

Feature	Description
Data Studio Updates	The Data Studio is updated with new features and changes in the following areas: • Data Load Dashboard: A new Data Load Dashboard is available which displays summaries of the recent load jobs and their status and view details about the data you load. See The Data Load Page for more information. • Sentiment Analysis and Key Phrase Extraction on text-based columns: You can now perform Sentiment Analysis and Key Phrase Extraction on the Data Load tool to analyze text-based data. See Use OCI Language Service Capabilities in Data Studio for more information. • Data Load support to access Amazon Web Services (AWS) Glue Catalogs: The Data Load tool of Data Studio allows you to synchronize Autonomous Database with Amazon Web Service (AWS) Glue Data Catalog metadata. See Register AWS Glue Catalog for more information. • Data Load support for registration and synchronization of OCI Data Catalogs: The Data Load tool enables you to register and synchronize OCI Data Catalogs to create external tables in Autonomous Database. See Register OCI Data Catalog for more information. • Data Load support to load GeoJSON files: The Data Load tool supports loading GeoJSON files in Autonomous Database. See Use GeoJSON in Data Load for more information. • Data Load support to enable loading into other schemas: The Data Load tool allows an ADMIN to create and load data into other schemas. See Loading Data From Local Files for more information. • Enable Resource Principal using Data Studio: You can enable Resource Principals using Data Studio to authenticate and access Oracle Cloud Infrastructure resources. See Manage Resource Principal using Data Studio for more information.
Manage Oracle Cloud Operator Access	Oracle Cloud Operators do not have authorization to access your data or any other information in your database schemas. When access to your database schemas is required to troubleshoot or mitigate an issue, you can allow a cloud operator to access your Autonomous Database schemas for a limited time. See Manage Oracle Cloud Operator Access for more information.
Apache Iceberg External Tables	Autonomous Database supports querying of Apache Iceberg tables stored in Amazon Web Services (AWS) or in Oracle Cloud Infrastructure Object Storage. See Query Apache Iceberg Tables for more information.

Feature	Description
Cloud Tables	You can create Cloud Tables where table data resides on Oracle managed Cloud Storage and the table data does not consume database storage. See Use Cloud Tables to Store Logging and Diagnostic Information for more information.
Cloud Links Offload Targets Without Specifying Database IDs	Cloud Links support with the optional `offload_targets` parameter lets you specify that access is offloaded to refreshable clones. This parameter now supports the `ANY` keyword, where all data set requests from consumers are offloaded to a refreshable clone. See Register a Data Set with Offload Targets for Data Set Access for more information.
Live Data Sharing Using Database Actions in Autonomous Database	Use Data Sharing to share data, as of the latest database commit, with Autonomous Databases in the same region. A Data Sharing recipient always views the latest data. See Overview of the Data Share Tool for more information.
Updated Version of Oracle Autonomous Database Free Container Image	A new version of the Oracle Autonomous Database Free Container Image is available with updates and new features. Use the Oracle Autonomous Database Free Container Image to run Autonomous Database in a container in your own environment, without requiring access to Oracle Cloud Infrastructure Console or to the internet. See Use the Oracle Autonomous Database Free Container Image for more information.

March 2024

Feature	Description
Refreshable Clone Automatic Refresh Option	When you enable the automatic refresh option Autonomous Database automatically refreshes a refreshable clone with data from the source database. See Refreshable Clone with Automatic Refresh Enabled and Edit Automatic Refresh Policy for Refreshable Clone for more information.
Invoke External Procedures	You can invoke external procedures written in the C language from Autonomous Database as SQL functions. You do not install external procedures on Autonomous Database. To use an external procedure, the procedure is hosted remotely on a VM running in an Oracle Cloud Infrastructure Virtual Cloud Network (VCN). See Invoke External Procedures as SQL Functions for more information.
Network File System (NFS) NFSv4 Support	You can attach a Network File System to a directory location in your Autonomous Database. This allows you to load data from Oracle Cloud Infrastructure File Storage in your Virtual Cloud Network (VCN) or from any other Network File System in on-premises data centers. Depending on the version of the Network File System you want to access, both NFSv3 and NFSv4 are supported. See Access Network File System from Autonomous Database for more information.
Support for Regular Expressions in DBMS_CLOUD Procedures	The format option `regexuri` is supported with the following `DBMS_CLOUD` procedures: `COPY_COLLECTION`, `COPY_DATA`, `CREATE_EXTERNAL_TABLE`, `CREATE_EXTERNAL_PART_TABLE`, and `CREATE_HYBRID_PART_TABLE`. See DBMS_CLOUD Package Format Options for more information

Feature	Description
Encrypt or Decrypt Data	For greater security you can encrypt data that you export to Object Storage. When data on Object Storage is encrypted you can decrypt the data you import or when you use the data in an external table. See Encrypt Data While Exporting or Decrypt Data While Importing for more information.
`JOB_QUEUE_PROCCESSES` Initialization Parameter	You can set the `JOB_QUEUE_PROCESSES` initialization parameter. Setting this parameter to 0 disables non-Oracle supplied Scheduler jobs. See JOB_QUEUE_PROCESSES for more information.
Pre-Authenticated Request URLs Access Information	You can generate and manage Pre-Authenticated Request (PAR) URLs for data on Autonomous Database and Autonomous Database provides views that allow you to monitor PAR URL usage. See Monitor and View PAR URL Usage for more information.
`UTL_HTTP` allows `HTTP_PROXY` Connections	With `UTL_HTTP`, when the Autonomous Database instance is on a private endpoint you can call `UTL_HTTP.set_proxy` and both `HTTPS` and `HTTP_PROXY` connections are allowed. See Submit an HTTP Request to a Private Host and PL/SQL Packages Notes for Autonomous Database for more information.

February 2024

Feature	Description
Pre-Authenticated Request URLs for Read Only Data Access	You can generate and manage Pre-Authenticated Request (PAR) URLs for data on Autonomous Database. See Use Pre-Authenticated Request URLs for Read Only Data Access on Autonomous Database for more information.
Replicate Backups to a Cross-Region Disaster Recovery Peer	You can enable replication of automatic backups from a primary database to a cross-region disaster recovery peer. Up to seven days of automatic backups are available in the remote region when this feature is enabled. See Replicating Backups to a Cross-Region Autonomous Data Guard Standby and Replicating Backups to a Cross-Region Backup Based Disaster Recovery Peer for more information.
Select AI with OCI Generative AI	Autonomous Database can interact with AI service providers. Select AI now supports OCI Generative AI, Azure OpenAI Service, OpenAI, and CohereAI. This feature supports having LLMs work with Oracle database by generating SQL from natural language prompts. This allows you to talk to your database. See Use Select AI to Generate SQL from Natural Language Prompts for more information.

January 2024

Feature	Description
ECPU Compute Model for all Workload Types	Autonomous Database offers two compute models when you create or clone an instance: ECPU and OCPU. Previously only the workload types Data Warehouse and Transaction Processing supported the ECPU compute model. Now all workload types support the ECPU compute model, including JSON and APEX. See Compute Models in Autonomous Database for more information.
Manage Time Zone File Updates	Autonomous Database provides several options to automatically update time zone files. See Manage Time Zone File Updates on Autonomous Database for more information.
Custom Kerberos Service Name Support for Kerberos Authentication	You can use a custom Kerberos service name when you configure Kerberos to authenticate Autonomous Database users. Using a custom service name you can use the same Keytab files on multiple Autonomous Database instances when you enable Kerberos authentication (you do not need to create and upload different Keytab files for each Autonomous Database instance). See Configure Kerberos Authentication with Autonomous Database for more information.
Clone Database to Change Workload Type	You can create a clone of an Autonomous Database and select a different workload type for the cloned database. See Clone Autonomous Database to Change Workload Type for more information.

About Autonomous Database

Provides information about Autonomous Database features, workload types, and regions.

- What is Oracle Autonomous Database?
  Oracle Autonomous Database provides an easy-to-use, fully autonomous database that scales elastically and delivers fast query performance. As a service, Autonomous Database does not require database administration.
- Key Features of Autonomous Database
  Provides information on key features of Autonomous Database, an affordable, feature-rich service in the cloud.
- About Autonomous Database Workload Types
  Autonomous Database supports different workload types, including: Data Warehouse, Transaction Processing, JSON Database, and APEX Service. Each of these workload types provides performance improvements and additional features that support operations for the specified workload.
- Autonomous Database Region Availability
  Describes availability information for Autonomous Database in commercial regions and government cloud regions.

What is Oracle Autonomous Database?

Oracle Autonomous Database provides an easy-to-use, fully autonomous database that scales elastically and delivers fast query performance. As a service, Autonomous Database does not require database administration.

With Autonomous Database you do not need to configure or manage any hardware or install any software. Autonomous Database handles provisioning the database, backing up the database, patching and upgrading the database, and growing or shrinking the database. Autonomous Database is a completely elastic service.

At any time you can scale, increase or decrease, either the compute or the storage capacity. When you make resource changes for your Autonomous Database instance, the resources automatically shrink or grow without requiring any downtime or service interruptions.

Autonomous Database is built upon Oracle Database, so that the applications and tools that support Oracle Database also support Autonomous Database. These tools and applications connect to Autonomous Database using standard SQL*Net connections. The tools and applications can either be in your data center or in a public cloud. Oracle Analytics Cloud and other Oracle Cloud services provide support for Autonomous Database connections.

Autonomous Database also includes the following:

- Oracle APEX: a low-code development platform that enables you to build scalable, secure enterprise apps with world-class features.
- Oracle REST Data Services (ORDS): a Java Enterprise Edition based data service that makes it easy to develop modern REST interfaces for relational data and JSON Document Store.
- Database Actions: is a web-based interface that uses Oracle REST Data Services to provide development, data tools, administration, and monitoring features for Autonomous Database.
- Oracle Machine Learning Notebooks Early Adopter is an enhanced web-based notebook platform for data engineers, data analysts, R and Python users, and data scientists. You can write code, text, create visualizations, and perform data analytics including machine learning. In Oracle Machine Learning, notebooks are available in a project within a workspace, where you can create, edit, delete, copy, move, and even save notebooks as templates.

Key Features of Autonomous Database

Provides information on key features of Autonomous Database, an affordable, feature-rich service in the cloud.

Key Features

- **Managed**: Oracle simplifies end-to-end management of the database:
  - Provisioning new databases
  - Growing or shrinking storage and compute resources
  - Patching and upgrades
  - Backup and recovery
- **Fully elastic scaling**: Scale compute and storage independently to fit your database workload with no downtime:

  - Size the database to the exact compute and storage required
  - Scale the database on demand: Independently scale compute or storage
  - Shut off idle compute to save money
- **Auto scaling**: Allows your database to use more CPU and IO resources or to use additional storage automatically when the workload or storage demand requires additional resources:
  - Specify the number of CPUs for your Autonomous Database workload.
  - Use compute auto scaling to allow the database to use up to three times more CPU and IO resources, depending on workload requirements. Compute auto scaling is enabled by default when you create an Autonomous Database.
  - Use storage auto scaling to allow the database to expand to use up to three times the reserved base storage, depending on your storage requirements. Storage auto scaling is disabled by default when you create an Autonomous Database.
  - Manage auto scaling from the Oracle Cloud Infrastructure Console to enable or disable compute auto scaling or storage auto scaling for your Autonomous Database.
- **Autonomous Database supports**:
  - Existing applications, running in the cloud or on-premise
  - Connectivity via SQL*Net, JDBC, ODBC
  - Third-party data-integration tools
  - Oracle cloud services: Oracle Analytics Cloud, Oracle GoldenGate Marketplace, and others
- **High-performance queries and concurrent workloads**: Optimized query performance with preconfigured resource profiles for different types of users.
- **Oracle SQL**: Autonomous Database is compatible with existing applications that support Oracle Database.
- **Built-in web-based data analysis tool**: Web-based notebook tool for designing and sharing SQL based data-driven, interactive documents.
- **Database migration utility**: Easily migrate from MySQL, Amazon AWS Redshift, PostgreSQL, SQL Server, and other databases.

Simple Cloud-based Data Loading

Autonomous Database provides:

- Fast, scalable data-loading from Oracle Cloud Infrastructure Object Storage, Azure Blob Storage, Amazon S3, Amazon S3-Compatible, GitHub Repository, Google Cloud Storage, or on-premise data sources.

Oracle Database Actions

Database Actions is a web-based interface that uses Oracle REST Data Services to provide development, data tools, and administration and monitoring features for Autonomous Database, including the following:

- **Development Tools**
  - SQL Navigator and Worksheet: view objects and enter and run SQL and PL/SQL statements, and create database objects

  - Data Modeler: provides an integrated version of Oracle SQL Developer Data Modeler with basic reporting features. You can create diagrams from existing schemas, retrieve data dictionary information, generate DDL statements, and export diagrams
  - REST: An IDE for your REST APIs that enables you to manage templates, handlers and OAuth clients, generate API documentation, and test APIs.
  - LIQUIBASE: View ChangeLogs applied to your schema.
  - JSON: Create collections, upload documents, query and filter your data, create diagrams for your JSON document structures, and create relational views and indexes.
  - Charts: Use SQL queries to build rich charts and dashboards containing multiple charts.
  - Scheduling: An interface for `DBMS_SCHEDULER` that enables you to monitor jobs, view execution history, forecast upcoming jobs, and visualize scheduler chains.
  - Oracle Machine Learning: provides several components accessible through a common user interface. OML Notebooks supports Python, SQL, PL/SQL, and Markdown interpreters, with access to in-database ML through OML4Py and OML4SQL. OML Models supports managing and deploying in-database models. OML AutoML UI provides a no-code user interface to build, evaluate, and deploy in-database models using automated machine learning.
  - APEX: Login to APEX, develop and run rich, low-code web applications.
  - Graph Studio: Oracle Graph Studio lets you create property graph databases and automates the creation of graph models and in-memory graphs from database tables.
- **Data Studio**
  - Data Load: load or access data from local files or remote databases.
  - Catalog: understand data dependencies and the impact of changes.
  - Data Insights: discover anomalies, outliers and hidden patterns in your data.
  - Data Analysis: analyze your data
  - Data Transforms: transform data for analysis and other applications.
- **Administration and Monitoring**
  - Manage users
  - Database Dashboard: Monitor database activity charts such as CPU usage, number of executing SQL statements, and wait events formerly found on your Autonomous Database Service Console.
  - Performance Hub: Access SQL Monitoring reports and Active Session History (ASH) Analytics.

Disaster Recovery Options

Autonomous Data Guard Autonomous Database provides Autonomous Data Guard to enable a standby (peer) database to provide data protection and disaster recovery for your Autonomous Database instance.When you add an Autonomous Data Guard standby database, the system creates a standby database that continuously gets updated with the changes from the primary database. You can use Autonomous Data Guard with a standby in the current region, a local standby, or with a standby in a different region, a cross-region standby. You can also use Autonomous Data Guard with both a local standby and a cross-region standby.

Backup-Based Disaster Recovery uses backups to instantiate a peer database at the time of switchover or failover. This enables you to have a lower cost and higher Recovery Time

Objective (RTO) disaster recovery option for your Autonomous Database, as compared with Autonomous Data Guard. For local backup-based disaster recovery, existing local backups are utilized. There are no additional costs for a local Backup-Based Disaster Recovery. Cross-Region Backup-Based Disaster Recovery incurs an additional cost.

SQL Developer Support

Using Autonomous Database with SQL Developer you can do the following:

- Connect to Autonomous Database
- Create tables, indexes, and materialized views in Autonomous Database
- Load data into an Autonomous Database
- Copy tables to Autonomous Database
- Transfer a schema to Autonomous Database

Business Intelligence Tools Support

Autonomous Database is compatible with a number of business intelligence and data visualization tools from Oracle and from trusted third parties:

- Oracle Analytics Cloud
- Oracle Analytics Desktop
- Third-party Business Intelligence tools

About Autonomous Database Workload Types

Autonomous Database supports different workload types, including: Data Warehouse, Transaction Processing, JSON Database, and APEX Service. Each of these workload types provides performance improvements and additional features that support operations for the specified workload.

- About Oracle Autonomous Data Warehouse
  Autonomous Database is designed as a "load and go" service: you start the service, define tables, load data, and then run queries.
- About Oracle Autonomous Transaction Processing
  Autonomous Database is designed to support all standard business applications and deliver scalable query performance.
- About Autonomous JSON Database
  Oracle Autonomous JSON Database is Oracle Autonomous Transaction Processing, but designed for developing NoSQL-style applications that use JavaScript Object Notation (JSON) documents. You can promote an Autonomous JSON Database service to an Autonomous Transaction Processing service.
- About Oracle APEX Application Development
  Oracle APEX Application Development (APEX Service) is a low cost, Oracle Cloud service offering convenient access to the Oracle APEX platform for rapidly building and deploying low-code applications. APEX Service is designed to support all standard business applications and deliver scalable query performance.

About Oracle Autonomous Data Warehouse

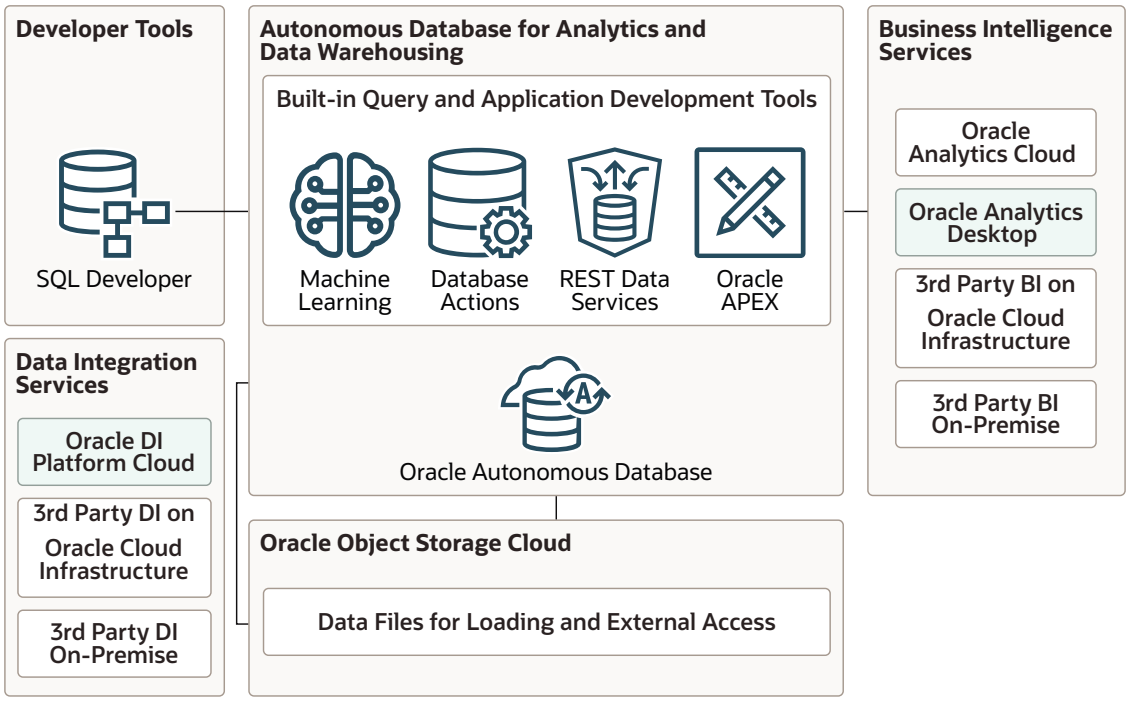
Autonomous Database is designed as a "load and go" service: you start the service, define tables, load data, and then run queries.

Autonomous Database is designed to support all standard SQL and business intelligence (BI) tools, and provides all of the performance of the market-leading Oracle Database in an environment that is tuned and optimized for data warehouse workloads.

To get started you create an Autonomous Database with workload type Data Warehouse and specify the number of ECPUs (OCPUs if your database uses OCPUs) and the storage capacity in TB's for the Autonomous Database.

You can use Autonomous Database with Oracle Analytics Cloud or Oracle Analytics Desktop to easily create visualizations and projects that reveal trends in your company's data and help you answer questions and discover important insights about your business.

The following figure shows the Autonomous Database architecture with related components for analytics and data warehousing.



About Oracle Autonomous Transaction Processing

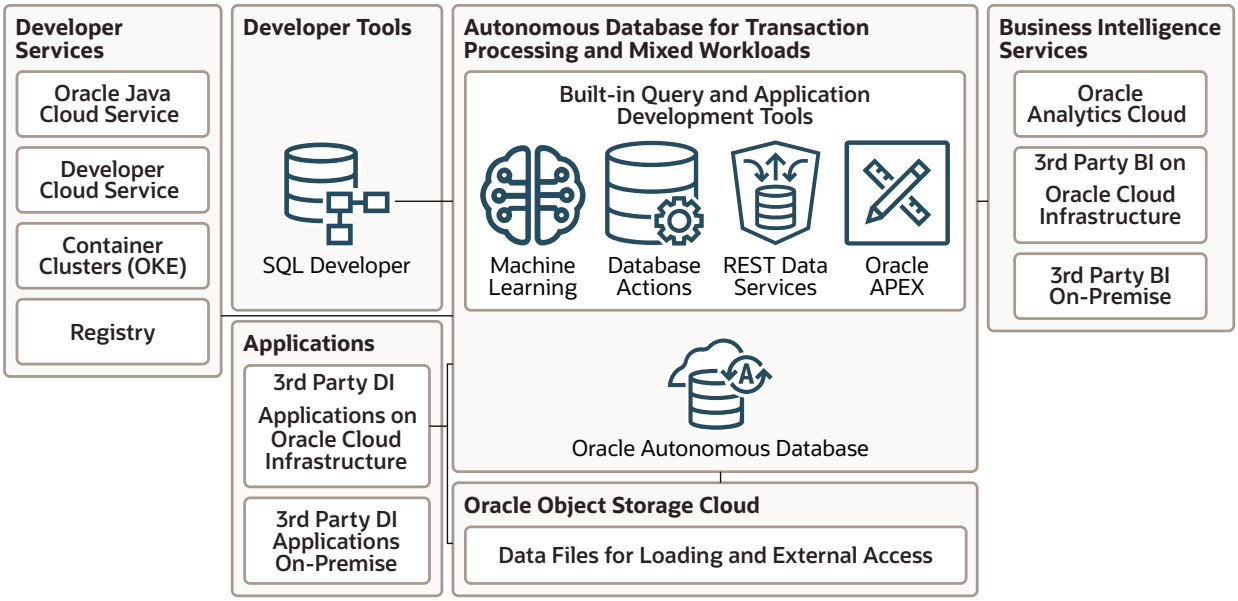
Autonomous Database is designed to support all standard business applications and deliver scalable query performance.

Autonomous Database provides all of the performance of the market-leading Oracle Database in an environment that is tuned and optimized to meet the demands of a variety of applications, including: mission-critical transaction processing, mixed transactions and analytics, IoT, and JSON document store.

To get started you create an Autonomous Database with the workload type Transaction Processing and specify the number of ECPUs (OCPUs if your database uses OCPUs) and the storage capacity in TB's for the database.

You can use Autonomous Database with Oracle Analytics Cloud or Oracle Analytics Desktop to easily create visualizations and projects that reveal trends in your company's operational data and help you answer questions and discover important insights about your business.

The following figure shows the Autonomous Database architecture with related components for transaction processing and mixed workloads.



About Autonomous JSON Database

Oracle Autonomous JSON Database is Oracle Autonomous Transaction Processing, but designed for developing NoSQL-style applications that use JavaScript Object Notation (JSON) documents. You can promote an Autonomous JSON Database service to an Autonomous Transaction Processing service.

Oracle Autonomous JSON Database provides all of the same features as Autonomous Transaction Processing, with this *important limitation*: you can store only up to 20 GB of data other than JSON document collections. There is no storage limit for JSON collections.

Development of NoSQL-style, document-centric applications is particularly flexible because the applications use *schemaless* data. This lets you quickly react to changing application requirements. There's no need to normalize the data into relational tables, and no impediment to changing data structure or organization at any time, in any way. A JSON document has internal structure, but no relation is imposed on separate JSON documents.

With Oracle Autonomous JSON Database your JSON document-centric applications typically use Simple Oracle Document Access (SODA), which is a set of NoSQL-style APIs for various application-development languages and for the representational state transfer (REST) architectural style. You can use any SODA API to access any SODA collection.

SODA document collections are backed by ordinary database tables and views. To use other kinds of data, subject to the 20 GB limit, you typically need some knowledge of Structured Query Language (SQL) and how that data is stored in the database.

With Oracle Autonomous JSON Database, a SODA collection *can only contain JSON data*. For example, you cannot have a collection of image documents or a collection that contains both JSON documents and image documents. This is a limitation relative to Autonomous Transaction Processing, where you can define such heterogeneous collections.

No matter what kind of data your applications use, whether JSON or something else, you can take advantage of all Oracle Database features. This is true regardless of the kind of Oracle Autonomous Database you use.

JSON data is stored natively in the database. In a SODA collection on an Autonomous Database JSON data is stored in Oracle's native binary format, OSON.

About Oracle APEX Application Development

Oracle APEX Application Development (APEX Service) is a low cost, Oracle Cloud service offering convenient access to the Oracle APEX platform for rapidly building and deploying low-code applications. APEX Service is designed to support all standard business applications and deliver scalable query performance.

See Oracle APEX Application Development for more information.

Autonomous Database Region Availability

Describes availability information for Autonomous Database in commercial regions and government cloud regions.

Commercial Regions and Availability Domains

Autonomous Database is available in all regions of the commercial realm.

Government Cloud

See Oracle Cloud Infrastructure Government Cloud for information about availability in Government Cloud regions.

Get Help, Search Forums, and Contact Support

When you use Oracle Autonomous Database, sometimes you need to get help from the community or to talk to someone in Oracle support. This topic provides more information about getting help by viewing and posting questions on forums and using Oracle Cloud Support to create a support request.

- Post Questions on Forums
  If you can't find an answer to a question through search, you can submit a question to one of the forums. This option is available to all customers.
- Open a Support Ticket
  If forums and other search options do not resolve your issue and you need to talk to someone, you can create a support request.

Post Questions on Forums

If you can't find an answer to a question through search, you can submit a question to one of the forums. This option is available to all customers.

Cloud Customer Connect Forums

For any issue related to Autonomous Database or Oracle Cloud Infrastructure, you can post a question to Cloud Customer Connect:

- Autonomous Data Warehouse
- Autonomous Transaction Processing

- Oracle Cloud Infrastructure and Platform

Stack Overflow Knowledge Forum

If you are working with Autonomous Database and you have technical questions, you can use stackoverflow to post questions and to find answers or to help others answer their questions. When you post, tag your question with oracle-autonomous-db, as follows:

Questions tagged [oracle-autonomous-db]

Open a Support Ticket

If forums and other search options do not resolve your issue and you need to talk to someone, you can create a support request.

If you need to file a service request use Oracle Cloud Support or contact your support representative and provide the tenancy details. In addition to support for technical issues, you can open support requests if you need to:

Use the Oracle Cloud Infrastructure Console to Create a Support Ticket

The first time you open a support ticket, you're automatically taken through a series of steps to provision your support account. If you want to make changes or if you run into problems, see Configuring Your Oracle Support Account.

To create a support request from the Oracle Cloud Infrastructure Console:

1. On the Oracle Cloud Infrastructure open the **Help** menu (?) and under **Request Help**, click **Create Support Request**.
2. Enter the following:
   - **Issue Summary**: Enter a title that summarizes your issue. Avoid entering confidential information.
   - **Describe Your Issue**: Provide a brief overview of your issue.
     - Include all the information that support needs to route and respond to your request.
       See Obtain Tenancy Details for details on obtaining Autonomous Database information.
     - Include troubleshooting steps taken and any available test results.
   - Select the severity level for this request.
3. Click **Create Support Request.**

Service Level Objectives (SLOs)

Describes the Service Level Objectives (SLOs) for Oracle Autonomous Database Serverless.

- Recovery Time Objective and Recovery Point Objective
  Oracle Autonomous Database Serverless is engineered to return an application online following an unplanned outage or a planned maintenance activity within single-digit seconds.
- Built-In Tool Availability
  Oracle will use commercially reasonable efforts to have the following tools with a Monthly Uptime Percentage (Availability) objective as defined below, during the service commitment of a calendar month.

- Zero-Regression Service Level Objective
  Oracle Autonomous Database Serverless automatically applies patches on your database. Oracle provides a service level objective of zero regressions in your production database due to these patches.

Recovery Time Objective and Recovery Point Objective

Oracle Autonomous Database Serverless is engineered to return an application online following an unplanned outage or a planned maintenance activity within single-digit seconds.

The following table outlines the target Recovery Time Objective (RTO) SLOs for different failure events.

Event	Recovery Time Objective (RTO) Service Level Objective	Recovery Point Objective (RPO) Service Level Objective, Maximum Possible Data Loss
Events requiring failover to a standby database when Autonomous Data Guard is enabled, such as: • Data corruptions • Full database failures • Complete storage failures • Availability Domain or Region failures	With an Autonomous Data Guard standby: **Local Standby Database**: two (2) minutes **Cross-Region Standby Database:** fifteen (15) minutes See Autonomous Data Guard Recovery Time Objective (RTO) and Recovery Point Objective (RPO) for more information.	**Local Standby Database**: The maximum possible data loss objective is < 10 seconds **Cross-Region Standby Database:** The maximum possible data loss objective is 1 minute
Events requiring failover to a backup copy or restoring from a backup (when there is not an Autonomous Data Guard standby) such as: • Data corruptions • Full database failures • Complete storage failures • Availability Domain or Region failures	With Backup-Based Disaster Recovery, based on the size of the database: **Local backup copy**: 1 hour + (1 hour per 5 TB) **Cross-Region backup copy**: 1 hour + (1 hour per 5 TB) For example, a 5 TB database has a Recovery Time Objective (RTO) of up to 2 hours. See Backup-Based Disaster Recovery Recovery Time Objective (RTO) and Recovery Point Objective (RPO) for more information.	**Local backup copy**: The maximum possible data loss objective is < 10 seconds **Cross-Region backup copy**: The maximum possible data loss objective is 1 minute

Built-In Tool Availability

Oracle will use commercially reasonable efforts to have the following tools with a Monthly Uptime Percentage (Availability) objective as defined below, during the service commitment of a calendar month.

Oracle Autonomous Database Built-in Tool	Availability Service Level Objective
Oracle REST Data Services (ORDS)	99.95%
Oracle APEX (on Autonomous Database)	99.95%
Database Actions	99.95%

Oracle Autonomous Database Built-in Tool	Availability Service Level Objective
Graph Studio	99.95%
Oracle Machine Learning: • Oracle Machine Learning Notebooks • Oracle Machine Learning Services • Oracle Machine Learning for Python/R • Oracle Machine Learning AutoML User Interface	99.95%
Data Transforms	99.95%

The following terms apply to the Availability Service Level Objective for the Oracle Autonomous Database Built-in Tools listed in this table:

- “HTTP Error Rate” applies separately to each Database Built-in Tool and means the percentage value corresponds to: (i) the total number of failed HTTP calls made to the applicable tool with a status of “Bad Gateway” or “Service Unavailable” in a minute period during a calendar month, divided by (ii) the total number of HTTP calls made to the tool in a minute period.
- “Monthly Uptime Percentage” is calculated by subtracting from 100%, the average of the HTTP Error Rate for each minute period during the applicable calendar month.

Oracle Autonomous Database Built-in Tool	Availability Service Level Objective
Oracle Database API for MongoDB	99.95%

The following terms apply to the Availability Service Level Objective for Oracle Database API for MongoDB:

- A "Connection through Oracle Database API for MongoDB" is a direct connection established from any tool or application to the Cloud Service using Oracle Database API for MongoDB.
- “Monthly Uptime Percentage” is calculated by subtracting from 100%, the percentage of minutes during the calendar month in which the applicable Cloud Service was "Unavailable".
- “Unavailable” means a minute period when: (i) no connection through Oracle Database API for MongoDB is or can be established and (ii) all continuous attempts, at least five, to establish such a connection fail.

Zero-Regression Service Level Objective

Oracle Autonomous Database Serverless automatically applies patches on your database. Oracle provides a service level objective of zero regressions in your production database due to these patches.

Oracle Autonomous Database Serverless automatically applies patches containing bug and security fixes during the maintenance windows announced on your database console. When you provision your test or pre-prod database using the Early patch level option, you can test the patches on these instances before the patches are applied to your production database. If you see issues in your test or pre-prod database, you can file service requests to get the issues addressed before the patch is applied to your production database.

Oracle provides a service level objective of zero regressions in your production database. "Regression" in this document is described as issues introduced by patches or updates to the

Autonomous Database performed during the maintenance window announced on your database console.

After a patch is applied to your database on the Early patch level, if you find and report an issue on that database through a service request, Oracle will use commercially reasonable efforts to address the problem so that the same issue does not occur in your production database.

See Set the Patch Level for more information.

How Is Autonomous Database Billed?

Provides details for Autonomous Database billing.

- Autonomous Database Billing Summary
  You are billed for Autonomous Database usage according to the values of two parameters: **compute** and **storage**. You select values for these parameters when you provision or scale an Autonomous Database instance.
- Compute Models in Autonomous Database
- Oracle Autonomous Database Serverless Features Billing
- Oracle Autonomous Database Serverless Billing for Database Tools

Autonomous Database Billing Summary

You are billed for Autonomous Database usage according to the values of two parameters: **compute** and **storage**. You select values for these parameters when you provision or scale an Autonomous Database instance.

When you provision an instance you select a workload type based on the workloads you want to run on your database. The pricing for compute and storage resources is based on the workload type you select.

The workload types are:

- **Data Warehouse**: optimized for analytic workloads, including data marts, data warehouses, data lakes, and data lakehouses.
- **Transaction Processing**: optimized to run transactional, analytical, and batch workloads concurrently.
- **JSON**: a cloud document database service that makes it simple to develop JSON-centric applications.
- **APEX**: a fully managed, low-code application development platform for building and deploying modern cloud, mobile, and data-driven applications.

Depending on the workload type, you select an Autonomous Database billing model:

- **ECPU Billing Model**: (Recommended billing model) Based on the number of cores allocated from the pool of compute servers and storage servers.
- **OCPU Billing Model**: (Legacy billing model) Provides compute capacity equivalent of a physical core of a processor with hyper-threading enabled.

See the Autonomous Database Cloud Price List for up-to-date pricing for each workload type.

Use the Cloud Cost Estimator to estimate the monthly cost for an Autonomous Database.

Compute and Storage Billing Summary

ECPU Compute Model

Compute Billing

Compute billing is based on the value you specify for **ECPU Count** and on the additional features you use. For example, if you use additional compute resources with **Compute auto scaling** selected, you are billed for the additional compute resources.

- Data Warehouse workload: See Billing Information: Autonomous Data Warehouse ECPU Compute Model
- Transaction Processing workload: See Billing Information: Autonomous Transaction Processing ECPU Compute Model
- JSON workload: See Billing Information: Autonomous JSON Database ECPU Compute Model
- APEX workload: See Billing Information: APEX Service ECPU Compute Model

Storage Billing

Billing for database storage is based on the value you specify for **Storage** and based on the features you use that require additional storage. For example, you are billed for additional storage usage if you select **Storage auto scaling** and you use additional storage resources.

The storage for backups is billed in addition to database storage.

- Data Warehouse workloads:
  - Database storage is provisioned in Terabytes (TBs) and charged in TB increments. The minimum database storage size is 1 TB.
  - Backup storage is separately charged in TB increments.

  See Billing Information: Autonomous Data Warehouse ECPU Compute Model.
- Transaction Processing, JSON, or APEX workloads:
  - Database storage is provisioned in Gigabytes or Terabytes (GBs or TBs) and charged in Gigabyte increments. The minimum database storage size is 20 GB.
  - Backup storage is separately charged in GB increments.

  See Billing Information: Autonomous Data Warehouse ECPU Compute Model

Also see Oracle Autonomous Database Serverless Features Billing

OCPU Compute Model

Compute Billing

Compute billing is based on the value you specify for **OCPU Count** and on the additional features you use. For example, if you use additional compute resources with **Compute auto scaling** selected, you are billed for the additional compute resources.

- Data Warehouse workload: See Billing Information: Autonomous Data Warehouse OCPU Compute Model
- Transaction Processing, JSON, or APEX workloads: See Billing Information: Autonomous Transaction Processing OCPU Compute Model

Storage Billing

Billing for database storage is based on the value you specify for **Storage** and based on the features you use that require additional storage. For example, you are billed for additional storage usage if you select **Storage auto scaling** and you use additional storage resources.

- For all workload types database storage is provisioned in Terabytes (TBs) and charged in TB increments.
- For all workload types, the storage for automatic backups is included in the cost of database storage at no additional cost.

See OCPU Compute Model Billing Information and Oracle Autonomous Database Serverless Features Billing

Compute Models in Autonomous Database

Autonomous Database offers two compute models when you create or clone an instance: ECPU and OCPU.

- **ECPU**: An ECPU is an abstracted measure of compute resources. ECPUs are based on the number of cores elastically allocated from a pool of compute and storage servers.

  While provisioning a new database or cloning an existing database:

  – The CPU count defaults to 2 ECPUs.

  – For databases that need more than 2 ECPUs, you must specify the number of assigned ECPUs as an integer. For example, you cannot assign 3.5 ECPUs to a database. The next available number of ECPUs above 3 is 4.

- **OCPU**: An OCPU is a physical measure of compute resources. OCPUs are based on the physical core of a processor with hyper-threading enabled.

  > **Note:**
  >
  > OCPU is a legacy billing metric and has been retired for Autonomous Data Warehouse (**Data Warehouse** workload type) and Autonomous Transaction Processing (**Transaction Processing** workload type). Oracle recommends using ECPUs for all new and existing Autonomous Database deployments. See Oracle Support Document 2998742.1 for more information.

  While provisioning a new database or cloning an existing database:

  – The CPU count defaults to 1 OCPU.

  – For databases that need more than 1 OCPU, you must specify the number of assigned OCPUs as an integer. For example, you cannot assign 3.5 OCPUs to a database. The next available number of OCPUs above 3 is 4.

See Provision Autonomous Database and Clone an Autonomous Database Instance for more information.

ECPU Compute Model Billing Information

Provides billing information for Autonomous Transaction Processing, Autonomous Data Warehouse, Autonomous JSON Database, and APEX Service with the ECPU compute model. Cost and usage reports are available from the Oracle Cloud Infrastructure Console, under **Billing and Cost Management**.

Cost reports look like this: `reports/cost-csv/0001000000101010101-00001.csv.gz`. Download the zip file and extract to view the details.

Usage reports look like this: `reports/usage-csv/0001000001010101.csv.gz`. Download the zip file extract to view the details.

Billing Information: Autonomous Transaction Processing ECPU Compute Model

Shows billing information for Autonomous Transaction Processing with the ECPU compute model.

Resource Description	SKU Name	Part Number	Metric	Min. Increments Allowed	Cost and Usage Reports
ECPU usage (License Included)	Oracle Autonomous Transaction Processing – ECPU	B95702	ECPU per hour	1 ECPU Minimum number of ECPUs is 2	Cost report: in the `product/Description` column, find, "Oracle Autonomous Transaction Processing – ECPU" Usage report: in the `product/resource` column, find `PIC_ATP_ECPU_LI`

Resource Description	SKU Name	Part Number	Metric	Min. Increments Allowed	Cost and Usage Reports
ECPU usage (Bring your own license)	Oracle Autonomous Transaction Processing – ECPU – BYOL	B95704	ECPU per hour	1 ECPU Minimum number of ECPUs is 2	Cost report: in the `product/Description` column, find, "Oracle Autonomous Transaction Processing – ECPU – BYOL" Usage report: in the `product/resource` column, find `PIC_ATP_ECPU_BYOL`
Database Storage usage	Oracle Autonomous Database Storage for Transaction Processing	B95706	Gigabyte Storage Capacity Per Month	1 GB Minimum storage size is 20 GB	Cost report: in the `product/Description` column, find, "Oracle Autonomous Database Storage for Transaction Processing" Usage report: in the `product/resource` column, find `PIC_ADB_STORAGE_ATP`
Backup Storage usage	Oracle Autonomous Database Storage	B95754	Gigabyte Storage Capacity Per Month	1 GB Backups are billed per GB	Cost report: in the `product/Description` column, find, "Database Backup Storage" Usage report: in the `product/resource` column, find `PIC_ADB_STORAGE`

See Cost and Usage Reports Overview for details on the cost and usage reports.

Billing Information: Autonomous Data Warehouse ECPU Compute Model

Shows billing information for Autonomous Data Warehouse with the ECPU compute model.

Resource Description	SKU Name	Part Number	Metric	Min. Increments Allowed	Cost and Usage Reports
ECPU usage (License Included)	Oracle Autonomous Data Warehouse – ECPU	B95701	ECPU per hour	1 ECPU Minimum number of ECPUs is 2	Cost report: in the `product/Description` column, find, "Oracle Autonomous Data Warehouse – ECPU" Usage report: in the `product/resource` column, find `PIC_ADW_ECPU_LI`
ECPU usage (Bring your own license)	Oracle Autonomous Data Warehouse – ECPU – BYOL	B95703	ECPU per hour	1 ECPU Minimum number of ECPUs is 2	Cost report: in the `product/Description` column, find, "Oracle Autonomous Data Warehouse – ECPU – BYOL" Usage report: in the `product/resource` column, find `PIC_ADW_ECPU_BYOL`
Database Storage usage	Oracle Autonomous Database Storage	B95754	Gigabyte Storage Capacity Per Month	1 TB (1024 GB)	Cost report: in the `product/Description` column, find, "Oracle Autonomous Database Storage" Usage report: in the `product/resource` column, find `PIC_ADB_STORAGE`
Backup Storage usage	Oracle Autonomous Database Storage	B95754	Gigabyte Storage Capacity Per Month	1 GB Backups are billed per GB	Cost report: in the `product/Description` column, find, "Oracle Autonomous Database Backup Storage" Usage report: in the `product/resource` column, find `PIC_ADB_STORAGE`

See Cost and Usage Reports Overview for details on the cost and usage reports.

Billing Information: Autonomous JSON Database ECPU Compute Model

Shows billing information for Autonomous JSON Database with the ECPU compute model.

Resource Description	SKU Name	Part Number	Metric	Min. Increments Allowed	Cost and Usage Reports
ECPU usage (License Included)	Autonomous JSON Database – ECPU	B99708	ECPU per hour	1 ECPU Minimum number of ECPUs is 2	Cost report: in the `product/Description` column, find, "Autonomous JSON Database – ECPU" Usage report: in the `product/resource` column, find `PIC_AJD_ECPU_LI`
Database Storage usage	Oracle Autonomous Database Storage for Transaction Processing	B95706	Gigabyte Storage Capacity Per Month	1 GB Minimum storage size is 20 GB	Cost report: in the `product/Description` column, find, "Oracle Autonomous Database Storage for Transaction Processing" Usage report: in the `product/resource` column, find `PIC_ADB_STORAGE_ATP`
Backup Storage usage	Oracle Autonomous Database Storage	B95754	Gigabyte Storage Capacity Per Month	1 GB Backups are billed per GB	Cost report: in the `product/Description` column, find, "Database Backup Storage" Usage report: in the `product/resource` column, find `PIC_ADB_STORAGE`

See Cost and Usage Reports Overview for details on the cost and usage reports.

Billing Information: APEX Service ECPU Compute Model

Shows billing information for APEX Service with the ECPU compute model.

Resource Description	SKU Name	Part Number	Metric	Min. Increments Allowed	Cost and Usage Reports
ECPU usage (License Included)	APEX Service – ECPU	B99709	ECPU per hour	1 ECPU Minimum number of ECPUs is 2	Cost report: in the `product/Description` column, find, "APEX Service – ECPU" Usage report: in the `product/resource` column, find `PIC_APEX_ECPU_LI`
Database Storage usage	Oracle Autonomous Database Storage for Transaction Processing	B95706	Gigabyte Storage Capacity Per Month	1 GB Minimum storage size is 20 GB	Cost report: in the `product/Description` column, find, "Oracle Autonomous Database Storage for Transaction Processing" Usage report: in the `product/resource` column, find `PIC_ADB_STORAGE_ATP`
Backup Storage usage	Oracle Autonomous Database Storage	B95754	Gigabyte Storage Capacity Per Month	1 GB Backups are billed per GB	Cost report: in the `product/Description` column, find, "Database Backup Storage" Usage report: in the `product/resource` column, find `PIC_ADB_STORAGE`

See Cost and Usage Reports Overview for details on the cost and usage reports.

OCPU Compute Model Billing Information

Provides billing information for Autonomous Transaction Processing, Autonomous Data Warehouse, Autonomous JSON Database, and APEX Service with the OCPU compute model.

> **Note:**
>
> OCPU is a legacy billing metric and has been retired for Autonomous Data Warehouse (**Data Warehouse** workload type) and Autonomous Transaction Processing (**Transaction Processing** workload type). Oracle recommends using ECPUs for all new and existing Autonomous Database deployments. See Oracle Support Document 2998742.1 for more information.

- Billing Information: Autonomous Transaction Processing OCPU Compute Model
- Billing Information: Autonomous Data Warehouse OCPU Compute Model
- Billing Information: Autonomous JSON Database OCPU Compute Model
- Billing Information: APEX Service OCPU Compute Model

Billing Information: Autonomous Transaction Processing OCPU Compute Model

Shows billing information for Autonomous Transaction Processing with the OCPU compute model.

Resource Description	SKU Name	Part Number	Metric	Min. Increments Allowed	Cost and Usage Reports
OCPU usage (License Included)	Oracle Autonomous Transaction Processing	B90453	OCPU per hour	1 OCPU	Cost report: In the `product/Description` column, find "Oracle Autonomous Transaction Processing" Usage report: in the `product/resource` column, find `PIC_ATP_COMPUTE`
OCPU usage (Bring your own license)	Oracle Autonomous Transaction Processing – BYOL	B90454	OCPU per hour	1 OCPU	Cost report: Cost report: in the `product/Description` column, find, "Oracle Autonomous Transaction Processing – BYOL" Usage report: in the `product/resourceID` column, find `PIC_ATP_COMPUTE_BYOL`

Resource Description	SKU Name	Part Number	Metric	Min. Increments Allowed	Cost and Usage Reports
Database Storage usage	Oracle Autonomous Transaction Processing - Exadata Storage	B90455	Terabyte Storage Capacity Per Month	1 TB	Cost report: in the `product/Description` column, find, "Oracle Autonomous Transaction Processing - Exadata Storage" Usage report: in the `product/resource` column, find `PIC_ATP_EXADATA_STORAGE`

See Cost and Usage Reports Overview for details on the cost and usage reports.

Billing Information: Autonomous Data Warehouse OCPU Compute Model

Shows billing information for Autonomous Data Warehouse with the OCPU compute model.

Resource Description	SKU Name	Part Number	Metric	Min. Increments Allowed	Cost and Usage Reports
OCPU usage (License Included)	Oracle Autonomous Data Warehouse	B89040	OCPU per hour	1 OCPU	Cost report: in the `product/Description` column, find "Oracle Autonomous Data Warehouse" Usage report: in the `product/resource` column, find `PIC_ADWC_COMPUTE`
OCPU usage (Bring your own license)	Oracle Autonomous Data Warehouse – BYOL	B89039	OCPU per hour	1 OCPU	Cost report: in the `product/Description` column, find "Oracle Autonomous Data Warehouse – BYOL" Usage report: in the `product/resource` column, find `PIC_ADWC_COMPUTE_BYOL`

Resource Description	SKU Name	Part Number	Metric	Min. Increments Allowed	Cost and Usage Reports
Database Storage usage	Oracle Autonomous Data Warehouse - Exadata Storage	B89041	Terabyte Storage Capacity Per Month	1 TB	Cost report: in the `product/Description` column, find "Oracle Autonomous Data Warehouse - Exadata Storage" Usage report: in the `product/resource` column, find `PIC_ADWC_EXADATA_STORAGE`

See Cost and Usage Reports Overview for details on the cost and usage reports.

Billing Information: Autonomous JSON Database OCPU Compute Model

Shows billing information for Autonomous JSON Database with the OCPU compute model.

Resource Description	SKU Name	Part Number	Metric	Min. Increments Allowed	Cost and Usage Reports
OCPU usage (License Included)	Oracle Autonomous JSON Database	B92212	OCPU per hour	1 OCPU	Cost report: in the `product/Description` column, find "Oracle Autonomous JSON Database" Usage report: in the `product/resource` column, find `PIC_AJD_COMPUTE`
Database Storage usage	Oracle Autonomous Transaction Processing - Exadata Storage	B90453	Terabyte Storage Capacity Per Month	1 TB	Cost report: in the `product/Description` column, find "Oracle Autonomous JSON Database - Exadata Storage" Usage report: in the `product/resource` column, find `PIC_ATP_EXADATA_STORAGE`

> **Note:**
>
> In the OCPU model, 60 days of automatic backups are included in the cost of database storage and have no additional charge. Long-term backups are billed additionally, as database storage (Part Number: B90453).

See Cost and Usage Reports Overview for details on the cost and usage reports.

Billing Information: APEX Service OCPU Compute Model

Shows billing information for APEX Service with the OCPU compute model.

Resource Description	SKU Name	Part Number	Metric	Min. Increments Allowed	Cost and Usage Reports
OCPU usage (License Included)	Oracle APEX Application Development	B92911	OCPU per hour	1 OCPU	Cost report: In the `product/Description` column, find "Oracle APEX Application Development" Usage report: in the `product/resource` column, find `PIC_ADBS_APEX_LI`
Database Storage usage	Oracle Autonomous Transaction Processing - Exadata Storage	B90453	Terabyte Storage Capacity Per Month	1 TB	Cost report: in the `product/Description` column, find, "Oracle Autonomous Transaction Processing - Exadata Storage" Usage report: in the `product/resource` column, find `PIC_ATP_EXADATA_STORAGE`

> **Note:**
>
> In the OCPU model, 60 days of automatic backups are included in the cost of database storage and have no additional charge. Long-term backups are billed additionally, as database storage (Part Number: B90453).

See Cost and Usage Reports Overview for details on the cost and usage reports.

Oracle Autonomous Database Serverless Features Billing

Shows billing information for Autonomous Database features for ECPU and OCPU compute models.

ECPU Compute Model

- **Automatic Backups**: The storage for backups is billed, per GB, in addition to your selected database storage.

  For example, if backups are occupying 200 GB of storage you are billed for 200 GB of backup storage (in addition to the usage billed for your selected number of ECPUs and database storage). See ECPU Compute Model Billing Information for more detail about which SKUs are billed for backups.

- **Long-Term Backups**: The storage for long-term backups is billed, per GB, as backup storage, in addition to your database storage.

  For example, if automatic backups are currently occupying 200 GB and long-term backups are occupying 600 GB of storage, you are billed for 800 GB of backup storage, in addition to the usage billed for your selected ECPUs and database storage. See ECPU Compute Model Billing Information for more detail about which SKUs are billed for each workload type and backups.

- **Compute Auto-Scaling**: When compute auto scaling is enabled, your database may use and you may be billed for additional ECPU consumption as needed by your workload, up to three times (3x) the number of base ECPUs, as shown in the **ECPU count** on the Oracle Cloud Infrastructure Console.

  – Your billed ECPU usage per hour while your database is running is based on the base number of ECPUs you selected for your database, plus any additional ECPU usage due to auto-scaling.

  – A stopped Autonomous Database instance has zero ECPU usage.

  – ECPU usage is measured each second, in units of whole ECPUs and averaged across an hour. If your database is running for less than an hour or auto-scales for only part of an hour, it is billed per second for the average ECPU consumption, over the base ECPUs, during that hour. The minimum ECPU consumption is one minute.

  For example, if your database has ECPU count 4 with **Compute auto scaling** enabled:

  – Assume in hour one your database is Available for the entire hour and ECPU utilization is below 4 ECPUs. Your database will be billed for 4 ECPUs.

  – Assume in hour two your database is Available for the entire hour and ECPU utilization is below 4 ECPUs for 30 minutes, 50% of the hour, and auto scales to 8 ECPUs for 30 minutes (the other 50% of the hour). The usage for this period, for billing, is 6 ECPUs (based on the average per-second ECPU usage for hour two).

- **Storage Auto-Scaling**:

  – For storage usage below your reserved base storage, you are billed based on your base storage.

- After your allocated storage exceeds your reserved base storage, storage usage is billed based on your allocated storage rounded up to the nearest TB, in a given hour.

For example, if your reserved base storage is 4 TB, until your allocated storage exceeds 4 TB of storage, you are billed based on your base storage (4 TB). After you exceed 4 TB, storage is billed based on the allocated storage rounded up to the nearest TB, in a given hour. In this example, if the allocated storage grows over 4 TB in a given hour, say to 4.9 TB, you are billed for 5 TB of storage from that hour onward.

If you then delete 1 TB of data, your allocated storage remains at 4.9 TB and you are billed for 5 TB until you perform a shrink operation. When you perform a shrink operation, may be able to shrink /reduce your allocated storage back to 3.9 TB. After the shrink operation completes and your allocated storage (3.9 TB) is once again below your reserved base storage (4 TB), you will once again be billed for your reserved base storage of 4 TB. See Shrink Storage for more information.

- **Autonomous Data Guard Standby – Local (Same-Region)**

  Local Autonomous Data Guard peer databases incur the additional cost of the base ECPUs and the storage of the Primary database, including any auto-scaled storage usage, billed on the Primary database itself. Auto-scaled ECPUs of the Primary database are not billed for additionally on the local Autonomous Data Guard peer database. The number of base ECPUs is specified by the number of ECPUs as shown in the **ECPU count** on the Oracle Cloud Infrastructure Console.

  For example, if you enable a local Autonomous Data Guard peer on a source database with:

  - 2 (base) ECPUs with **Compute auto-scaling** enabled and are consuming about 4 ECPUs per hour
  - 1 TB of (base) storage with storage auto-scaling, consuming a total of 2 TB of database storage

  For the local Autonomous Data Guard peer you are billed for an additional 2 ECPUs (your base ECPU selection), plus an additional 2 TB of storage (that is, the same amount of storage reserved for the source Primary with auto-scale, on the Primary database).

  When the Primary database is stopped, neither the primary nor the peer database is billed for ECPU.

- **Autonomous Data Guard Standby – Remote (Cross-Region)**

  Autonomous Data Guard cross-region peer databases incur the additional cost of the base ECPUs and twice (2x) the storage of the Primary database, including any auto-scaled storage usage, billed on the remote peer database. Auto-scaled ECPUs of the Primary are not billed for additionally on the remote peer database. The number of base ECPUs is specified by the number of ECPUs as shown in the **ECPU count** on the Oracle Cloud Infrastructure Console.

  For example, if you enable a cross-region Autonomous Data Guard peer for a source database with:

  - 2 (base) ECPUs with **Compute auto-scaling** enabled and are consuming about 4 ECPUs per hour
  - 1 TB of (base) storage with storage auto-scaling, consuming a total of 2 TB of database storage

  For the cross-region Autonomous Data Guard peer you are billed an additional 2 ECPUs (your base ECPU selection), plus 4 TB of storage (that is 2x the storage reserved for the source Primary with auto-scaling, billed on the remote peer database).

When the Primary database is stopped, neither the primary nor the peer database are billed for ECPU.

When the option **Enable cross-region backup replication to disaster recovery peer** is selected, you are billed for twice (2x) the backup storage size required for the replicated backups, billed to the remote standby.

- **Backup-Based Disaster Recovery – Local (Same-Region) Backup Copies** There are no additional costs for local Backup-Based Disaster Recovery, other than the cost of storage for automatic backups.
- **Backup-Based Disaster Recovery – Remote (Cross-Region) Backup Copies** Billing for a cross-region Backup-Based Disaster Recovery is twice (2x) the amount of backup storage required for the replicated cross-region backups, billed to the remote peer.

  For example, if you enable a cross-region backup copy on a source database with:

  – 2 (base) ECPUs

  – 2 TB of database storage

  If the backups replicated to the remote region take up 1.9 TB of storage, you will be billed for 3.8 TB of backup storage on the remote backup copy peer database.

  When the option **Enable cross-region backup replication to disaster recovery peer** is selected, you are billed for twice (2x) the amount of backup storage size required for the additional replicated backups, billed to the remote peer. This billing is based on the number of days set for the backup retention on the Primary, as follows:

  – When automatic backup retention is set to 7 days or greater, billing is based on the storage size for 7 days of replicated backups.

  – When automatic backup retention is set to less than 7 days, billing is based on the storage size for the specified number of days of data that is replicated to the cross-region standby.
- **Refreshable Clone Local (Same-Region)** Local refreshable clones have their own configurable ECPU selection, and so they are billed for ECPU based on the user-selected number of ECPUs, with or without auto scaling; they do not get billed additionally over the ECPU selection. The number of ECPUs is specified by the number of ECPUs as shown in the **ECPU count** on the Oracle Cloud Infrastructure Console.

  Local refreshable clones are billed for the same amount of storage as their source database.

  For example, if you create a 2 ECPU local refreshable clone from a source database with:

  – 4 ECPUs

  – 1 TB of storage with storage auto-scaling and consuming 2 TB of storage

  For the local refreshable clone you will be billed for 2 ECPUs, that is, the refreshable clone's **ECPU count** value, and 2 TB of storage (that is, the storage reserved for the source database).

  When you start or stop the source database, your actions on the source database do not affect the refreshable clone. A refreshable clone is started or stopped independently of the source database.
- **Refreshable Clone Remote (Cross-Region)** Remote refreshable clones have their own configurable ECPU selection, and so they are billed for ECPU based on the user-selected ECPU (with or without auto scaling); they do not get billed additionally over the ECPU selection. The number of ECPUs is specified by the number of ECPUs as shown in the **ECPU count** on the Oracle Cloud Infrastructure Console.

Remote refreshable clones are billed for twice (2x) the amount of storage as their source database.

For example, if you create a 2 ECPU remote refreshable clone from a source database with:

- 4 ECPUs
- 1 TB of storage with storage auto-scaling and consuming 2 TB of storage

For the remote refreshable clone you are billed for 2 ECPUs (that is, the refreshable clone's ECPU selection) and 4 TB of storage (that is, 2x the storage reserved for the source database)

Starting or stopping the source database does not affect the refreshable clone - The refreshable clone can be started or stopped independently.

- **Snapshot Standby for Remote (Cross-Region) Disaster Recovery**

  Snapshot standby ECPU usage is billed based on the base ECPU count and any additional ECPU usage if compute auto-scaling is enabled. The number of base ECPUs is specified by the number of ECPUs, as shown in the **ECPU count** on the Oracle Cloud Infrastructure Console.

  Snapshot standby storage usage is billed based on the storage of the snapshot standby plus (1x) the storage of the source primary database.

  For example, if you have a 2 ECPU and 3 TB snapshot standby from a source database with:

  - 4 ECPUs
  - 1 TB of storage with storage auto-scaling and consuming 2 TB of storage

  Your snapshot standby will be billed for 2 ECPUs (that is, the snapshot standby's ECPU selection) and 3 TB + 2 TB = 5 TB of database storage (that is, the storage reserved on the snapshot standby + the storage reserved on its source database)

- **Elastic Pools**: An elastic pool allows you to provision 4 times the ECPU count that you have as your pool size. For example, if you have a pool with pool size of 128 ECPUs, you can provision up to 512 ECPUs in this pool. In other words, when your pool size is 128 ECPUs, your pool capacity is four times the pool size (in this example, 512 ECPUs).

  Databases that belong to a pool are not billed individually for compute. The compute billing for all the pool members and the leader happens through the leader. In other words, individual members of an elastic pool do not get billed for compute as long as they are part of the pool. This applies no matter what the workload type is for the pool member. For example, when a pool member with workload type **Data Warehouse** is added to a pool, it's compute usage is billed to the pool leader at the **Transaction Processing** compute usage rate. The storage billing, on the other hand, continues to be billed to individual Autonomous Database instances irrespective of them being part of a pool or not.

  Let’s assume you have an elastic pool with pool size of 128 ECPUs. Given the pool size, the pool capacity is 512 ECPUs (pool capacity = 4x pool size). For this sample, here are some common billing questions and answers:

  - What is the maximum number of Autonomous Database instances allowed in this pool? A total of 512 Autonomous Database instances with 1 ECPU each (elastic pool members or the leader can have an individual ECPU allocation as low as 1 ECPU).
  - What happens if the aggregated ECPU utilization high watermark of the pool is greater than the pool size? If the aggregated ECPU utilization high watermark is less than or equal to the pool size in a given billing hour, the hourly charge is in the amount of the pool size. If the aggregated ECPU utilization high watermark is more than the pool size but less than or equal to the 2x pool size in a given billing hour, the hourly charge is in

the amount of 2x pool size. If the aggregated ECPU utilization high watermark is more than 2x pool size in a given billing hour, the hourly charge is in the amount of 4x pool size.

For example, let's assume 512 Autonomous Database instances with 1 ECPU each are in an elastic pool with a pool size of 128 ECPUs. If the aggregated ECPU utilization high watermark of these databases is 100 ECPUs between 1pm-2pm, and 250 ECPUs between 2pm-3pm; then the billing is 128 ECPU hours between 1pm-2pm and 256 ECPU hours between 2pm-3pm.

See About Elastic Pool Billing for more information.

OCPU Compute Model

- **Automatic Backups**: The storage for automatic backups is included in the cost of database storage. See OCPU Compute Model Billing Information for more detail about which SKUs for database storage.
- **Long-Term Backups**: The storage for long-term backups is billed per TB as additional database storage, in addition to your selected database storage usage.

  For example, if automatic backups occupy 200 GB and long-term backups occupy 600 GB of storage, you will be billed for 1 TB (600 GB of long-term backup storage rounded up to the nearest TB) as database storage, in addition to the usage billed for your selected OCPU and database storage. See OCPU Compute Model Billing Information for more detail about which SKUs are billed for each workload type and backups.
- **Compute auto-scaling**: When compute auto scaling is enabled your database may use and you may be billed for additional OCPU consumption as needed by your workload, up to three times (3x) the number of base OCPUs (as shown in **OCPU count** on the Oracle Cloud Infrastructure Console)
  - Your billed OCPU usage per hour while your database is running is based on the base number of OCPUs you selected for your database, plus any additional OCPU usage due to auto-scaling.
  - A stopped Autonomous Database instance has zero OCPU usage.
  - OCPU usage is measured each second, in units of whole OCPUs and averaged across an hour. If your database is running for less than an hour or auto-scales for only part of an hour, it will be billed per second for the average OCPU consumption (over the base OCPU) during that hour. The minimum OCPU consumption is one minute.
- **Storage Auto-Scaling**:
  - For storage usage below your reserved base storage, you are billed based on your base storage.
  - After your allocated storage exceeds your reserved base storage, storage usage is billed based on your allocated storage rounded up to the nearest TB, in a given hour.

  For example, if your reserved base storage is 4 TB, until your allocated storage exceeds 4 TB of storage, you are billed based on your base storage (4 TB). After you exceed 4 TB, storage is billed based on the allocated storage rounded up to the nearest TB, in a given hour. In this example, if the allocated storage grows over 4 TB in a given hour, say to 4.9 TB, you are billed for 5 TB of storage from that hour onward.

  If you then delete 1 TB of data, your allocated storage remains at 4.9 TB and you are billed for 5 TB until you perform a shrink operation. When you perform a shrink operation, Autonomous Database may be able to shrink / reduce your allocated storage back to 3.9 TB. After the shrink operation completes and your allocated storage (3.9TB) is once again below your reserved base storage (4 TB), you will once again be billed for your reserved base storage of 4 TB. See Shrink Storage for more information.

- **Autonomous Data Guard Standby Local (same-region)**

  Local Autonomous Data Guard peer databases incur the additional cost of the base OCPUs and the storage of the Primary database, including any auto-scaled storage usage, billed on the Primary database itself. Auto-scaled OCPUs of the Primary database are not billed for additionally on the local Autonomous Data Guard peer database. The number of base OCPUs is specified by the number of OCPUs, as shown in the **OCPU count** on the Oracle Cloud Infrastructure Console.

  When the Primary database is stopped, neither the primary nor the peer database is billed for OCPU.

- **Autonomous Data Guard Standby Remote (cross-region)**

  Autonomous Data Guard cross-region standby databases incur the additional cost of the base OCPUs and twice (2x) the storage of the Primary database, including any auto-scaled storage usage, billed on the remote peer database. Auto-scaled OCPUs of the Primary are not billed for additionally on the remote peer database. The number of base OCPUs is specified by the number of OCPUs, as shown in **OCPU count** on the Oracle Cloud Infrastructure Console.

  When the Primary database is stopped, neither the primary nor the peer database is billed for OCPU.

  When the option **Enable cross-region backup replication to disaster recovery peer** is selected, you are billed to the remote standby database's OCPU database storage, for twice (2x) the 7 day replicated backup storage size, rounded up to the nearest TB.

- **Backup-based Disaster Recovery Local (same-region) Backup Copies** There are no additional costs for local Backup-Based Disaster Recovery, other than the cost of keeping automatic backups.

- **Backup-Based Disaster Recovery Remote (cross-region) Backup Copies**

  Billing for a cross-region Backup-Based Disaster Recovery with OCPUs is twice (2x) the amount of storage required for backups replicated to the remote region, billed as database storage to the remote peer, rounded up to the nearest TB.

  When the option **Enable cross-region backup replication to disaster recovery peer** is selected, you are billed to the remote peer database's OCPU database storage, for twice (2x) the replicated backup storage size, rounded up to the nearest TB.

- **Refreshable Clone Local (same-region)**

  Local refreshable clones have their own configurable OCPU selection, and so they are billed for OCPU based on the user-selected OCPU (with or without auto scaling); they do not get billed additionally over the OCPU selection. The number of OCPUs is specified by the number OCPUs as shown in the **OCPU count** on the Oracle Cloud Infrastructure Console.

  Local refreshable clones are billed for the same amount of storage as their source database.

  Starting or stopping the source database does not affect the refreshable clone. The refreshable clone can be started or stopped independently.

- **Refreshable Clone Remote (cross-region)**

  Remote refreshable clones have their own configurable OCPU selection, and so they are billed for OCPU based on the user-selected OCPU (with or without auto scaling); they do not get billed additionally over the OCPU selection. The number of OCPUs is specified by the number of OCPUs as shown in the **OCPU count** on the Oracle Cloud Infrastructure Console.

Remote refreshable clones are billed for twice (2x) the amount of storage as their source database.

Starting or stopping the source database does not affect the refreshable clone. The refreshable clone can be started or stopped independently.

- **Snapshot Standby for Remote (cross-region) disaster recovery**

  Snapshot standby OCPU usage is billed based on the base OCPU count and any additional OCPU usage if compute auto-scaling is enabled. The number of base OCPUs is specified by the OCPUs as shown in the **OCPU count** on the Oracle Cloud Infrastructure Console.

  Snapshot standby storage usage is billed based on the storage of the snapshot standby plus (1x) the storage of the source primary database.

Oracle Autonomous Database Serverless Billing for Database Tools

Shows billing information for Autonomous Database database tools for ECPU and OCPU compute models.

ECPU Compute Model

The following built-in tools have separate, configurable compute resource limits:

- Graph Studio
- Oracle Machine Learning
- Data Transforms

For these tools compute resource usage does not use the database's base compute, as specified with the **ECPU count**, or any auto-scaled ECPUs. These tools have separate ECPU allocation, independent from the database compute **ECPU count** (and independent of **Compute auto scaling**).

For Graph Studio, Oracle Machine Learning, and Data Transforms, the ECPU count and Maximum idle time configuration options let you specify compute resource allocation for the VMs that run the associated built-in tool. When you configure these built-in tools, the values for these configuration options mean the following in terms of resource usage and billing:

- You do not pay for a tool's ECPU allocation if you do not use the tool.
- The VMs associated with a tool are provisioned when you start using a tool. For example, if Graph Studio is disabled, billing does not begin when you enable the tool. Billing begins when you start using Graph Studio.
- The ECPU count specifies the compute resource allocation that is dedicated to the tool. The value you enter for ECPU count applies in addition to the database ECPU count you specify for the instance. After you start using a tool with a specified ECPU count, you are billed per ECPU hour reserved from the time the built-in tool is launched.

- Billing stops for a built-in tool's allocated ECPUs when the built-in tool is disabled, the instance stops or is terminated, or when the built-in tool is idle for more than the specified Maximum idle time. The default maximum idle times depend on the tool:
- The default maximum idle times depend on the built-in database tool:
  - Graph Studio: 240 minutes
  - Oracle Machine Learning User Interface: 60 minutes
  - Data Transforms: 10 minutes

For example, assume you allocate 4 ECPUs, as the base ECPU count, for the Autonomous Database instance, with compute auto-scaling disabled, and 2 ECPUs for Graph Studio and set the maximum idle time to 30 minutes. Using Graph Studio for 30 min between 2 pm and 3 pm, your bill for the specified time period (2-3pm) is 4 ECPUs plus the additional tool usage for the Graph Studio usage of 2 ECPUs, which is a total of 6 ECPUs, assuming no other tools are used in this period.

As another example, assume you allocate 6 ECPUs, as the base ECPU count, for the Autonomous Database instance, with compute auto-scaling disabled, and you configure 4 ECPUs for Oracle Machine Learning (OML) with the maximum idle time set to 15 minutes. When the database is open and available for the hour between 2 to 3 pm, and you use OML for 15 min starting at 2 pm, with the specified 15 minute maximum idle time, you are charged for 4 ECPUs for OML compute usage for 30 minutes plus 6 ECPUs for the database compute resources for the full hour, for a total of 8 ECPUs assuming no other tools are used in this period.

Note: billing is per minute.

OCPU Compute Model

Built-in tools do not have separate, configurable compute resource limits. The built-in tools use compute from the database's selected OCPU count. Tool usage may also use auto-scaled OCPUs, if necessary.

Always Free Autonomous Database

You have the option to create a limited number of Always Free Autonomous Databases that do not consume cloud credits. Always Free databases can be created in Oracle Cloud Infrastructure accounts that are in a trial period, have paying status, or are always free. This section describes configuration differences, restrictions, and additional details for Always Free databases.

Sign Up with Oracle Cloud Free Tier

These are the services you can use for an unlimited time:

- Two Oracle Autonomous Databases with powerful tools like Oracle APEX and Oracle SQL Developer
- Two Oracle Cloud Infrastructure Compute VMs; Block, Object, and Archive Storage; Load Balancer and data egress; Monitoring and Notifications

See Oracle Cloud Free Tier to start for free.

Resource Restrictions for Always Free Autonomous Database

- The Always Free option provides databases that have CPU and storage included and you are never billed for an Always Free instance until the instance is upgraded to a paid Autonomous Database.
- Maximum of approximately 20 GB Exadata storage per database (you may see more than this)
- Maximum of 30 simultaneous database sessions
- Maximum of 2 Always Free Autonomous Database instances per Oracle Cloud Infrastructure tenancy. The Always Free Autonomous Database workload types are: Data Warehouse, Transaction Processing, JSON Database, and APEX Service. If you create 2 Always Free instances, they can be the same or different Autonomous Database workload types.
- The HTTP interface for Always Free Autonomous Databases is rate limited to restrict the number of simultaneous service users. Approximately 3-6 simultaneous users can be supported across all of the APEX, Oracle REST Data Services, and Database Actions running on your Always Free Autonomous Databases. Additional simultaneous users beyond that may result in users encountering HTTP errors such as HTTP status code 429.

  This HTTP interface rate limit applies only for Always Free Autonomous Databases.

> **Notes:**
>
> - For details on Always Free Oracle APEX Application Development (APEX Service), see Always Free Oracle APEX Application Development.
> - Always Free Autonomous Databases cannot be scaled manually or automatically beyond the fixed resource restrictions described above.
> - The Maximum of 30 simultaneous database sessions limit for Always Free allows you to work with Autonomous Database; however, if your usage includes many simultaneous users and/or many concurrent database client connections then you can exceed these limits, resulting in errors. To avoid such errors, obtain more resources for your Autonomous Database by upgrading to paid service.
> - Always Free Autonomous Databases cannot be provisioned as a private endpoint and cannot reside within a Virtual Cloud Network (VCN). See Configure Private Endpoints When You Provision or Clone an Instance for further information on private endpoints.

Oracle Database Version

The available Database versions for Always Free Autonomous Database are:

- **Oracle Database 19c**
- **Oracle Database 23ai**

  If you are using Always Free Autonomous Database with Oracle Database 23ai, then many of the concepts and features of this service are further documented here:

  – Oracle Database 23ai

  – *Oracle Database New Features*

> **Note:**
>
> Always Free Autonomous Database with Oracle Database 23ai is available in the following regions: Ashburn, Phoenix, Frankfurt, and London. To take advantage of new features available with Oracle Database 23ai, your tenancy must have a home region of one of those regions and you must provision your Always Free Autonomous Database in your home region.

See Always Free Autonomous Database Oracle Database 23ai Notes for additional information and limitations on Always Free Autonomous Database with Oracle Database 23ai.

Regional Availability for Always Free Autonomous Database

- Always Free Autonomous Databases are available worldwide in a subset of Oracle Cloud Infrastructure data regions. See Data Regions for more details on where Always Free databases are supported.
- When you sign up for Oracle Cloud Infrastructure, Oracle creates a tenancy and designates a home data region for the tenancy that you specify. You can create Always Free Autonomous Databases only in this home data region. You cannot create an Always Free Autonomous Database in other data regions that you subsequently subscribe to. See The Home Region for more information.

Backup Functionality Not Available in Always Free Autonomous Database

- Always Free Autonomous Databases do not support long-term backups or manual backups to your Oracle Cloud Infrastructure object storage.
- Always Free Autonomous Databases do not support restore (you cannot restore the current database to the selected past backup or timestamp).
- To use backup and restore functionality on Autonomous Database you must upgrade an Always Free Autonomous Database to a paid instance.

See Backup and Restore Autonomous Database Instances and Upgrade an Always Free Autonomous Database to a Paid Instance for more information.

Autonomous Data Guard Not Available for Always Free Autonomous Database

Autonomous Data Guard is not available with Always Free Autonomous Databases. See Use Standby Databases with Autonomous Data Guard for Disaster Recovery for more information.

Tools Configuration Options Not Available for Always Free Autonomous Database

Always Free Autonomous Database does not provide configuration options for Autonomous Database tools and does not allow you to disable Autonomous Database tools. For example, you cannot specifically disable HTTP access to Oracle APEX and REST Data Services on Always Free databases.

Supplemental Logging and Oracle GoldenGate Extract are Not Available for Always Free Autonomous Database

The following are not available for Always Free Autonomous Databases:

- Supplemental logging
- Oracle GoldenGate Extract

See Supplemental Logging and Configuring Extract to Capture from an Autonomous Database for more information.

Inactivity Monitoring and Database Stoppage

Persistently inactive Always Free Autonomous Databases are detected and handled as follows:

- After being inactive for 7 days, the database will be stopped automatically, preserving its stored data. Inactivity measurements leading up to 7 days are based on database connections and CPU usage. Successfully making a SQL*Net or HTTPS connection and running SQL commands on your database resets these measurements to zero.
- A database that is automatically or manually stopped and stays inactive for 90 days, cumulative, may be reclaimed and permanently deleted. Inactivity measurements leading up to 90 days are based on the database being inactive or in the stopped state. Starting a stopped database resets these measurements to zero.

  Start an Always Free Autonomous Database by clicking the **Start** button on the Oracle Cloud Infrastructure console. Start a stopped Always Free Autonomous Database before 90 days to avoid losing access to its data.

  When you start an Always Free Autonomous Database from the stopped state, you need to wait about 5 minutes before attempting to connect to an APEX application or to an Oracle REST Data Services (ORDS) endpoint. If you attempt to connect before the background APEX and ORDS startup completes, then you may see HTTP error messages.
- On an Always Free database the Oracle Cloud Infrastructure console shows banner alerts prior to automatic stop and permanent delete operations occurring. If you subscribe to Oracle Cloud Infrastructure Alerts and Notifications, you also will receive email notifications.

> **Note:**
>
> After an Always Free Autonomous Database has been stopped and is later started, you may need to reconnect to the database from SQL*Net database clients. You can use the same Oracle Wallet and database user credentials to reconnect.

- Upgrade an Always Free Autonomous Database to a Paid Instance
  You can upgrade Always Free Autonomous Databases to paid instances to give them additional compute and storage resources.

Upgrade an Always Free Autonomous Database to a Paid Instance

You can upgrade Always Free Autonomous Databases to paid instances to give them additional compute and storage resources.

> **Note:**
>
> Promotion of Always Free to a paid Autonomous Database is supported only if the Database version for the Always Free Autonomous Database is Oracle Database 19c.

For details on upgrading your Always Free APEX Service to an Oracle APEX Application Development paid instance, see Upgrading Always Free APEX Service to a Paid Version.

If your account has finished a trial without upgrading to paying status, you can continue using Always Free databases but you cannot upgrade Always Free instances to paid instances until the account is first upgraded to paying status. See Upgrade Your Free Oracle Cloud Promotion for more information.

If your Oracle Cloud Infrastructure account is in a trial period or has paying status and the Oracle Database version for the Always Free database is Oracle Database 19c, then you can upgrade the Always Free database to a paid instance as follows:

1. From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
2. On the Autonomous Databases page select an Always Free Autonomous Database from the links under the **Display name** column.
3. On the Autonomous Database Details page, from the **More actions** drop-down list select **Upgrade instance to paid**.
4. Click **Upgrade instance to paid**.

When you upgrade from an Always Free instance to a paid instance you get a paid instance of the same workload type with the minimum CPU and minimum storage available for that workload type. After you upgrade you can scale up CPU and storage resources to fit your needs.

For example:

- **Data Warehouse** workload type: When you upgrade to paid from an Always Free instance, the upgrade process provides a paid Autonomous Database instance with workload type **Data Warehouse**, with 2 ECPUs and 1 TB of database storage.
- **Transaction Processing** workload type: When you upgrade to paid from an Always Free instance, the upgrade process provides a paid Autonomous Database instance with workload type **Transaction Processing**, with 2 ECPUs and 1 TB of database storage.

2
Quick Start

- Before You Begin with Oracle Autonomous Database
  Before you begin using Oracle Autonomous Database, you should be familiar with Oracle Cloud.
- Autonomous Database 15 Minute Quick Start
  Learn about Autonomous Database Serverless and learn how to create an Autonomous Database in just a few clicks.

Before You Begin with Oracle Autonomous Database

Before you begin using Oracle Autonomous Database, you should be familiar with Oracle Cloud.

Before you create an Autonomous Database:

- On Oracle Cloud, sign up for Oracle Cloud Free Tier or sign up for a paid Cloud Account. You cannot create an Autonomous Database deployment until you do so.

  See Welcome to Oracle Cloud Infrastructure for more information.
- (Optional) if you want to leverage an object store for data loading you need your object store credentials to use with Oracle Autonomous Database, including a *username* and a *password*. For details on the required credentials, depending on the object store you want to use, see the following:
  - **Oracle Cloud Infrastructure Object Storage**, the `username` is your Oracle Cloud Infrastructure user name. The `password` is your auth token. See Working with Auth Tokens.
  - **Oracle Cloud Infrastructure Object Storage Classic**, the `username` is your Oracle Cloud Infrastructure Classic user name and the `password` is your Oracle Cloud Infrastructure Classic password.
  - **Amazon S3**, the `username` is your AWS access key ID and the `password` is your AWS secret access key. See AWS Identity and Access Management.
  - **Azure Blob Storage**, the `username` is your Azure storage account name and the `password` is an Azure storage account access key. See About Azure storage accounts.
  - **Amazon S3-Compatible**, such as Oracle Cloud Infrastructure Object Storage, Google Cloud Storage, and Wasabi Hot Cloud Storage. See CREATE_CREDENTIAL Procedure for more information.
  - **GitHub Repository**, the `username` is your GitHub email and the `password` is your GitHub personal access token. See Creating a personal access token for more information.

Autonomous Database 15 Minute Quick Start

Learn about Autonomous Database Serverless and learn how to create an Autonomous Database in just a few clicks.

 Autonomous Database 15 Minute Quick Start Workshop

- Deploy an Autonomous Database instance that is optimized for data warehousing workloads.
- Use Autonomous Database tools to load object storage sources.
- Use advanced SQL to uncover issues and possibilities

3 Tasks

- Provision or Clone an Autonomous Database
  Provisioning a new Autonomous Database is easy. You can also provision a new Autonomous Database by cloning an existing instance.
- Connect to Autonomous Database
  Describes methods to securely connect to Autonomous Database.
- Manage Users
  Describes administration tasks for managing database users on Autonomous Database.
- Manage Credentials or Configure Policies and Roles to Access Resources
  Describes the methods in which Autonomous Database connects to other resources, either with credentials, or by configuring policies and roles for access.
- Load Data into Autonomous Database
  Describes packages and tools to load data with Autonomous Database.
- Link Data
  You can link to data in remote databases, in cloud storage, or in connected file systems or the local file system. You can also use Database Links to access objects on another database.
- Call Web Services from Autonomous Database
  Describes options for calling Web Services from Autonomous Database.
- Move Files
  Describes how to create directories and copy files to Autonomous Database.
- Replicate Data
  Describes using Oracle GoldenGate to replicate data between an Autonomous Database instance and to any target database or platform that Oracle GoldenGate supports.
- Exporting Data from Autonomous Database to Object Store or to Other Oracle Databases
  To export data from an Autonomous Database, use one of the following methods:
- Data Sharing
  Share data on Autonomous Database
- Develop
  Describes developer tasks and procedures on Autonomous Database.
- Analyze
  Provides information on the built-in tools that allow you to explore your data with reports, analytics, and visualization.
- Manage the Service
  Describes basic tasks for managing an Autonomous Database instance.
- Managing and Viewing Maintenance Windows, Patching Options, Work Requests, and Customer Contacts
- Monitor and Manage Performance
  Describes the Database Dashboard card in Database Actions where you can find information about the performance of an Autonomous Database instance.

- Use Sample Data Sets in Autonomous Database
  For users who want to start using the service without creating their own tables, Autonomous Database provides the read-only Sales History and Star Schema Benchmark data sets.
- Use the Oracle Autonomous Database Free Container Image

Provision or Clone an Autonomous Database

Provisioning a new Autonomous Database is easy. You can also provision a new Autonomous Database by cloning an existing instance.

See Provision Autonomous Database for details on how to create an Autonomous Database for your workload type using the Create Autonomous Database dialog.

See Clone an Autonomous Database Instance for details on cloning.

To get started, you can also try the LiveLabs Autonomous Database 15 Minute Quick Start.

After you create an Autonomous Database:

- (Optional) Reset the administrator password:

  When you create an Autonomous Database you are required to set the administrator password. If you want to change the administrator password or if you need to unlock the administrator account, see Manage the Administrator Account on Autonomous Database.
- (Optional) Add Oracle Machine Learning component users. See Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database for more information.
- (Optional) Download Oracle Analytics Desktop and add a connection to Autonomous Database. See Create Dashboards and Reports to Analyze and Visualize Your Data.

Connect to Autonomous Database

Describes methods to securely connect to Autonomous Database.

- About Connecting to an Autonomous Database Instance
  After you create database users, applications and tools connect to Autonomous Databases using Oracle Net Services (also known as SQL*Net). Oracle Net Services enables a network session from a client application to an Oracle Database server.
- Connect to Autonomous Database Using a Client Application
  Autonomous Database is preconfigured to support Oracle Net Services (a TNS listener is installed and configured to use secure TCPS).
- Download Database Connection Information
- Connect to Autonomous Database Using Oracle Database Tools
  Oracle Database Tools such as SQL Developer, SQL*Plus, and SQLcl can be used with Autonomous Database.
- Connect with Built-In Oracle Database Actions
  You can access Database Actions from Autonomous Database. Database Actions provides development tools, data tools, administration, and monitoring features for Autonomous Database. Using Database Actions you can run SQL statements, queries, and scripts in a worksheet.

- Connect with JDBC Thin Driver
  Autonomous Database mandates a secure connection that uses Transport Layer Security (TLSv1.2).
- Preparing for Oracle Call Interface Connections
  Using TLS with Oracle Call Interface connections with newer Oracle Instant Client/Oracle Database Client versions, a wallet is not required. For TLS connections with Oracle Call Interface running on older Oracle Instant Client/Oracle Database Client versions, a wallet is required for both mTLS authenticated connections and TLS authenticated connections.
- Predefined Database Service Names for Autonomous Database
  The `tnsnames.ora` file provided with the credentials zip file contains the database service names that allow you to connect to your database.
- Connect with Oracle Analytics Desktop
  Oracle Analytics Desktop makes it easy to visualize your data so you can focus on exploring interesting data patterns. Just connect to Autonomous Database, select the elements that you're interested in, and let Oracle Analytics Desktop find the best way to visualize it. Choose from a variety of visualizations to look at data in a specific way.
- Connect with Oracle Analytics Cloud
  Oracle Analytics Cloud is a scalable and secure public cloud service that provides a full set of capabilities to explore and perform collaborative analytics for you, your workgroup, and your enterprise.
- Connection and Networking Options and Features
  Autonomous Database provides a number of different connection and networking options and features for connecting to a database.

About Connecting to an Autonomous Database Instance

After you create database users, applications and tools connect to Autonomous Databases using Oracle Net Services (also known as SQL*Net). Oracle Net Services enables a network session from a client application to an Oracle Database server.

When a network session is established, Oracle Net Services acts as the data courier for both the client application and the database. It is responsible for establishing and maintaining the connection between the client application and the database, as well as exchanging messages between them.

Oracle Net Services supports a variety of connection types to connect to an Autonomous Database instance, including:

- **JDBC Thin Driver**: for Java applications, the JDBC Thin Driver is a pure Java driver. Many applications, including Oracle SQL Developer support JDBC Thin Driver connections.
- **JDBC OCI**: which is used by Java language applications. JDBC OCI adds a layer over Oracle Call Interface for Java applications. The Oracle SQLcl command-line interface uses JDBC OCI.
- **Oracle Call Interface (OCI)**: used by many applications written in C language. Examples that use Oracle Call Interface include Oracle utilities such as Oracle SQL*Plus, SQL*Loader, and Oracle Data Pump.
- **ODBC Drivers**: used by applications running on Microsoft Windows, that are layered over Oracle Call Interface (OCI).

Third-party products and custom applications may use any of these connection types.

- Secure Connections to Autonomous Database
  Connections to Autonomous Database are made either over the public Internet, optionally with Access Control Rules (ACLs) defined, or using a private endpoint inside a Virtual Cloud Network (VCN) in your tenancy. When you specify a private endpoint configuration, this only allows traffic from the virtual cloud network you specify and blocks access to the database from all public IPs or VCNs. Configuring a private endpoint allows you to keep all traffic to and from your database off of the public internet.
- Connect to Autonomous Database Through a Firewall
  Most organizations protect networks and devices on a network using a firewall. A firewall controls incoming and outgoing network traffic using rules which allow the use of certain ports and access to certain computers (or, more specifically IP addresses or host names). An important function of a firewall is to provide separation between internal networks and the public internet.
- Using Application Continuity
  Application Continuity masks outages from end users and applications by recovering the in-flight work for impacted database sessions following outages. Application Continuity performs this recovery beneath the application so that the outage appears to the application as a slightly delayed execution.

Secure Connections to Autonomous Database

Connections to Autonomous Database are made either over the public Internet, optionally with Access Control Rules (ACLs) defined, or using a private endpoint inside a Virtual Cloud Network (VCN) in your tenancy. When you specify a private endpoint configuration, this only allows traffic from the virtual cloud network you specify and blocks access to the database from all public IPs or VCNs. Configuring a private endpoint allows you to keep all traffic to and from your database off of the public internet.

Many applications provide support for more than one connection type, but each type of connection to Autonomous Database uses certificate authentication and TCPS (Secure TCP) database connection using standard TLS 1.2. This ensures that there is no unauthorized access to the Autonomous Database and that communications between the client and server are fully encrypted and cannot be intercepted or altered.

Autonomous Database by default supports Mutual TLS (mTLS) connections. You have the option to configure an Autonomous Database instance to support both mTLS and TLS connections.

There are advantages for clients using TLS authentication with Autonomous Database, including the following:

- TLS connections do not require that you download a wallet. For TLS connections using JDBC Thin Driver with JDK8 or higher, a wallet is not required. This includes connections coming from the clients such as SQL Developer and SQL Command Line (SQLcl).
- Clients connecting with TLS do not need to worry about wallet rotation. Wallet rotation is a regular procedure for mTLS connections.
- TLS connections can be faster (providing less connection latency). TLS authentication can provide reduced connection latency compared to mTLS.
- TLS and mTLS connections are not mutually exclusive. Mutual TLS (mTLS) authentication is enabled by default and always available. When you enable TLS authentication, you can use either mTLS or TLS authentication.
- Using TLS authentication does not compromise the fully encrypted end-to-end communication between a client and Autonomous Database.

Topics

- About Mutual TLS (mTLS) Authentication
  Using Mutual Transport Layer Security (mTLS), clients connect through a TCPS (Secure TCP) database connection using standard TLS 1.2 with a trusted client certificate authority (CA) certificate. With mutual authentication both the client application and Autonomous Database authenticate each other. Autonomous Database uses mTLS authentication by default.
- About TLS Authentication
  Using Transport Layer Security (TLS), clients connect through a TCPS (Secure TCP) database connection using standard TLS 1.2. A client uses its list of trusted Certificate Authorities (CA)s to validate the server's CA root certificate. If the issuing CA is trusted, the client verifies that the certificate is authentic. This allows the client and Autonomous Database to establish the encrypted connection before exchanging any messages.

About Mutual TLS (mTLS) Authentication

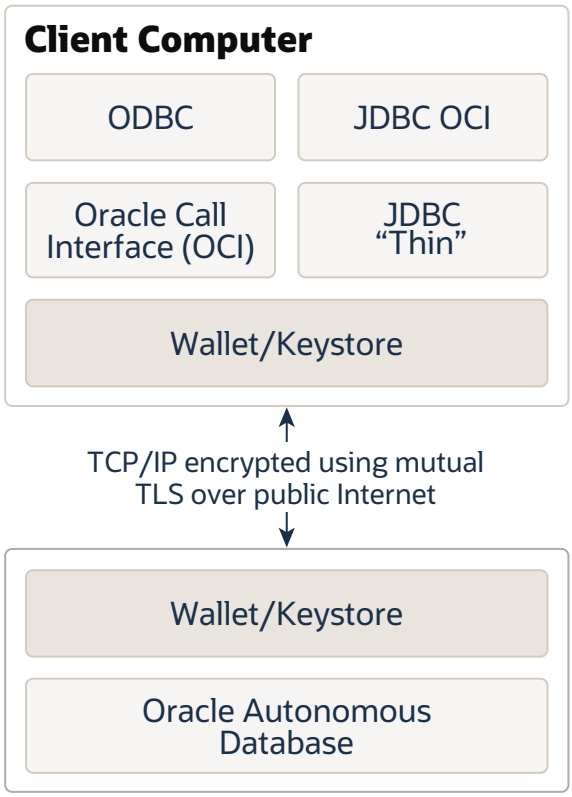
Using Mutual Transport Layer Security (mTLS), clients connect through a TCPS (Secure TCP) database connection using standard TLS 1.2 with a trusted client certificate authority (CA) certificate. With mutual authentication both the client application and Autonomous Database authenticate each other. Autonomous Database uses mTLS authentication by default.

Mutual TLS authentication requires that the client downloads or obtains a trusted client CA certificate for connecting to an Autonomous Database instance. Autonomous Database then uses the certificate to authenticate the client. This provides increased security and specifies the clients that can communicate with an Autonomous Database instance.

Certification authentication with Mutual TLS uses an encrypted key stored in a *wallet* on both the client (where the application is running) and the server (where your database service on the Autonomous Database is running). The key on the client must match the key on the server to make a connection. A wallet contains a collection of files, including the key and other information needed to connect to your Autonomous Database instance. All communications between the client and the server are encrypted.

To secure the connection to your Autonomous Database instance a service administrator downloads the client credentials (wallet files) from Autonomous Database. If you are not an Autonomous Database service administrator, your administrator provides you with the client credentials. See Download Client Credentials (Wallets) for more information.

The following figure shows client secure connections to Oracle Autonomous Database over the public Internet using Mutual TLS connections. If you configure your database to use private endpoints, then the public internet is not used and the connection uses a private endpoint inside a Virtual Cloud Network (VCN) in your tenancy.



About TLS Authentication

Using Transport Layer Security (TLS), clients connect through a TCPS (Secure TCP) database connection using standard TLS 1.2. A client uses its list of trusted Certificate Authorities (CA)s to validate the server's CA root certificate. If the issuing CA is trusted, the client verifies that the certificate is authentic. This allows the client and Autonomous Database to establish the encrypted connection before exchanging any messages.

When you connect with TLS authentication using JDBC Thin Driver clients, including Oracle SQL Developer and Oracle SQLcl, you do not need to download a wallet to secure the connection to your Autonomous Database instance. TLS authentication enables the client to verify the identity of the Autonomous Database service to provide secure communication.

Depending on the type of client, a TLS connection has the following support with Autonomous Database:

- For connections with JDBC Thin Driver using JDK8u162 or higher, including connections with Oracle SQL Developer and Oracle SQLcl, a wallet is not required.
- Oracle Call Interface (OCI) clients support TLS authentication without a wallet if you are using the following client versions:
  - Oracle Instant Client/Oracle Database Client 19.13 - only on Linux x64
  - Oracle Instant Client/Oracle Database Client 19.14 (or later) and 21.5 (or later) - only on Linux x64 and Windows
- If the client is connecting with managed ODP.NET or ODP.NET Core versions 19.13 or 21.4 (or above) using TLS authentication, the client can connect without providing a wallet.

There are network access prerequisites for TLS connections. See Network Access Prerequisites for TLS Connections for more information.

Connect to Autonomous Database Through a Firewall

Most organizations protect networks and devices on a network using a firewall. A firewall controls incoming and outgoing network traffic using rules which allow the use of certain ports and access to certain computers (or, more specifically IP addresses or host names). An important function of a firewall is to provide separation between internal networks and the public internet.

When Autonomous Database is configured for access using the public internet, you must configure the firewall to allow access to Autonomous Database servers.

To access the Autonomous Database from behind a firewall, the firewall must permit the use of the port specified in the database connection when connecting to the servers in the connection. The default port number for Autonomous Database mTLS connections is 1522 (find the port number in the connection string from the `tnsnames.ora` file in your `credentials ZIP` file). For example, see the `port` value in the following `tnsnames.ora` file:

```
db2022adb_high = (description = (
                address=(protocol=tcps)
                (port=1522)
                (host=adb.example.oraclecloud.com))
                (connect_data=(service_name=example_high.adb.oraclecloud.com))
                (security=(ssl_server_dn_match=yes)))
```

Your firewall must allow access to servers within the `.oraclecloud.com` domain using port 1522. To connect to Autonomous Database, depending upon your organization's network configuration, you may need to use a proxy server to access this port or you may need to request that your network administrator open this port.

Using Application Continuity

Application Continuity masks outages from end users and applications by recovering the in-flight work for impacted database sessions following outages. Application Continuity performs this recovery beneath the application so that the outage appears to the application as a slightly delayed execution.

> **Note:**
>
> By default Application Continuity is disabled.

See Use Application Continuity on Autonomous Database for more information on Application Continuity.

Connect to Autonomous Database Using a Client Application

Autonomous Database is preconfigured to support Oracle Net Services (a TNS listener is installed and configured to use secure TCPS).

The client computer must be prepared to use Oracle Net Services to connect to Autonomous Database.

- About Connecting to Autonomous Database Using a Client Application
  Applications can connect to Autonomous Database using any of the connection types supported by Oracle Net Services.
- Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections with Wallets (mTLS)
  Preparing for any type of Oracle Call Interface (OCI) connection with mTLS authentication requires the installation of client software, downloading client credentials, and configuring certain files and environment variables.
- Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections Using TLS Authentication
  Preparing for any type of Oracle Call Interface (OCI) connection with TLS authentication requires the installation of client software and configuring certain files and environment variables.
- Prepare for JDBC Thin Connections
  Applications that use JDBC Thin connections include the software necessary to make an Oracle Net Services connection. It is not necessary to download and install Oracle Client software.
- Connect Microsoft .NET, Visual Studio Code, and Visual Studio with a Wallet (mTLS)
  Oracle Autonomous Database supports connectivity to the Microsoft .NET Framework, .NET Core, Visual Studio, and Visual Studio Code.
- Connect Microsoft .NET, Visual Studio Code, and Visual Studio Without a Wallet
  Oracle Autonomous Database supports connectivity to the Microsoft .NET Framework, .NET Core, Visual Studio, and Visual Studio Code using TLS authentication without a wallet.
- Connect Node.js and other Scripting Languages (mTLS)
  You can use programs in different languages, including Python, Node.js, PHP, Ruby, R, Go, and Perl to connect to an Autonomous Database instance using mTLS (with wallets). Security is enforced using client credentials.
- Connect Node.js, and Other Scripting Languages Without a Wallet
  You can use programs in different languages, including Python, Node.js, PHP, Ruby, R, Go, and Perl to connect to an Autonomous Database instance using TLS authentication without a wallet.
- Connect Power BI and Microsoft Data Tools to Autonomous Database
  Oracle Client for Microsoft Tools (OCMT) is a graphical user interface (GUI) native Microsoft Software installer (MSI) that simplifies ODP.NET setup and Oracle database connectivity to multiple Microsoft data tools.
- Connect Python Applications to Autonomous Database
  You can connect Python applications to your Autonomous Database instance either with a wallet (mTLS) or without a wallet (TLS).

About Connecting to Autonomous Database Using a Client Application

Applications can connect to Autonomous Database using any of the connection types supported by Oracle Net Services.

Consult your application documentation for details about how your application connects to Oracle.

The following steps describe the process of connecting to Autonomous Database using a client application:

1. Determine what connection type your application uses, (for example OCI, ODBC, JDBC Thin, and so on).

2. The steps required to prepare the client computer depend on the type of connection the client application uses. Determine what authentication type your application uses, either mTLS or TLS and prepare your client computer:
   - **Mutual TLS authentication connections**: In all cases, with mTLS connections, client credentials in the form of a wallet file must be downloaded to the client.
   - **TLS authentication connections**: For connections with JDBC Thin Driver using JDK8u162 or higher, including connections with Oracle SQL Developer and Oracle SQLcl, a wallet is not required.

     Oracle Call Interface (OCI) clients support TLS authentication without a wallet if you are using the following client versions:
     - Oracle Instant Client/Oracle Database Client 19.13 - only on Linux x64
     - Oracle Instant Client/Oracle Database Client 19.14 (or later) and 21.5 (or later) - only on Linux x64 and Windows

   See the details for each driver type for information on the steps required to prepare your application to connect to Autonomous Database.
3. Within your application, set up the connection.

Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections with Wallets (mTLS)

Preparing for any type of Oracle Call Interface (OCI) connection with mTLS authentication requires the installation of client software, downloading client credentials, and configuring certain files and environment variables.

This topic covers the steps to prepare an application to connect using mTLS authentication with a wallet that you download from an Autonomous Database instance. See Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections Using TLS Authentication for information on the steps to prepare for TLS authentication with these connection types.

New Oracle Client Installation

The following steps assume Oracle client software has not already been installed on the client computer. If Oracle client software has already been installed and there are working copies of `sqlnet.ora` and `tnsnames.ora`, see Updating an Existing Oracle Client Installation.

Before making an Oracle Call Interface (OCI), ODBC, or JDBC OCI connection, do the following:

1. Install Oracle Client software on your computer.
   - **Oracle Instant Client/Oracle Database Client**: 18.19 (or later), 19.2 (or later), or 21 (base release or later). The Instant Client contains the minimal software needed to make an Oracle Call Interface connection.
2. Download client credentials and store the file in a secure folder on your client computer. See Download Client Credentials (Wallets).
3. Unzip/uncompress the credentials file into a secure folder on your client computer.
4. Edit the `sqlnet.ora` file in the folder where you unzip the credentials file, replacing `"?/network/admin"` with the name of the folder containing the client credentials.

For example, edit `sqlnet.ora` as follows:

```
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA = (DIRECTORY="?/
network/admin")))
SSL_SERVER_DN_MATCH=yes
```

The changed value on UNIX/Linux is:

```
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA = (DIRECTORY="/
home/adb_credentials")))
SSL_SERVER_DN_MATCH=yes
```

The changed value for Windows is:

```
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA = (DIRECTORY="D:\
\myapp\\adb_credentials")))
SSL_SERVER_DN_MATCH=yes
```

5. Create the `TNS_ADMIN` environment variable and set it to the location of the credentials file.

   Use this environment variable to change the directory path of Oracle Net Services configuration files from the default location of `ORACLE_HOME\network\admin` to the location of the secure folder containing the credentials file you saved in Step 2. Set the `TNS_ADMIN` environment variable to the directory where the unzipped credentials files are, not to the credentials file itself.

   For example, on UNIX/Linux set `TNS_ADMIN` to the full path of the directory where you unzipped the client credentials:

   ```
   export TNS_ADMIN=/home/adb_credentials
   ```

   For example on Windows:

   ```
   set TNS_ADMIN=d:\myapp\adb_credentials
   ```

Connections with an HTTP Proxy

If the client is behind a firewall and your network configuration requires an HTTP proxy to connect to the internet, then perform the following steps to update the `sqlnet.ora` and `tnsnames.ora` files. Connections through an HTTP proxy are only available with Oracle Client software version 12.2.0.1 or later.

> **Note:**
>
> To avoid manual updates in `sqlnet.ora` and `tnsnames.ora` files, you can use SQLcl and specify the HTTP proxy on the command line. See Connect Oracle SQLcl Cloud with a Wallet (mTLS) for more information.

1. Add the following line to the `sqlnet.ora` file to enable connections through an HTTP proxy:

   ```
   SQLNET.USE_HTTPS_PROXY=on
   ```

2. Add the HTTP proxy hostname and port to the connection definitions in `tnsnames.ora`. You need to add the `https_proxy` and `https_proxy_port` parameters in the address section of connection definitions. For example, the following sets the HTTP proxy to `proxyhostname` and the HTTP proxy port to `80`; replace these values with your HTTP proxy information:

   ```
   ADB1_high =
          (description=
                (address=
                      (https_proxy=proxyhostname)(https_proxy_port=80)
   (protocol=tcps)(port=1522)(host=adb.example.oraclecloud.com)
                )
                (connect_data=(service_name=adb1_high.adb.oraclecloud.com)
                )
                (security=(ssl_server_dn_match=yes))
                )
          )
   ```

> **Note:**
>
> Configuring `sqlnet.ora` and `tnsnames.ora` for the HTTP proxy may not be enough depending on your organization's network configuration and security policies. For example, some networks require a `username` and `password` for the HTTP proxy. In such cases contact your network administrator to open outbound connections to hosts in the oraclecloud.com domain using port 1522 without going through an HTTP proxy.

For more information on `SQLNET.USE_HTTPS_PROXY`, see *Net Services Reference*.

For information on `HTTPS_PROXY` and `HTTPS_PROXY_PORT`, see Protocol Address Section.

Updating an Existing Oracle Client Installation

If you have an existing Oracle Client installation, you already have `sqlnet.ora` and `tnsnames.ora` files and the `TNS_ADMIN` environment variable. In this case, do the following:

1. Update your `sqlnet.ora` file by adding the following:

   ```
   WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA = (DIRECTORY="/
   home/adb_credentials")))
   ```

2. Copy the entries in the `tnsnames.ora` file provided in the Autonomous Database wallet to your existing `tnsnames.ora` file.

Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections Using TLS Authentication

Preparing for any type of Oracle Call Interface (OCI) connection with TLS authentication requires the installation of client software and configuring certain files and environment variables.

Oracle Call Interface (OCI) clients support TLS authentication without a wallet if you are using the following client versions:

- Oracle Instant Client/Oracle Database Client 19.13 - only on Linux x64
- Oracle Instant Client/Oracle Database Client 19.14 (or later) and 21.5 (or later) - only on Linux x64 and Windows

See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for information on allowing TLS connections.

1. Install Oracle Instant Client.
   a. Got to the Oracle Instant Client page and click **Download Now**: Oracle Instant Client
   b. On the Oracle Instant Client Downloads page, select your platform.

      For example, under **Instant Client for Linux**, select the Instant Client for Linux x86-64 architecture (for this example, to download the Linux x86-64 version).
   c. Under **Version 19.14.0.0.0 (Requires glibc 2.14)**, select an Instant Client package to download.
   d. If you are building a language API or driver from source code, you may also need to download the Instant Client SDK Package version 19.14: Oracle Instant Client
   e. Unzip the base package you selected. If you also download the SDK, unzip it in the same directory.
   f. On Linux, create a symbolic link if it does not exist. For example:

      ```
      cd /home/myuser/instantclient_19_14
      ln -s libclntsh.so.19.1 libclntsh.so
      ```

      If there is no other Oracle software on your system that will be impacted, add Instant Client to the runtime link path. For example:

      ```
      sudo sh -c "echo /home/myuser/instantclient_19_14 > /etc/ld.so.conf.d/
      oic.conf"
      sudo ldconfig
      ```

      Alternatively set the library path in each shell that runs your application. For example:

      ```
      export LD_LIBRARY_PATH=/home/myuser/instantclient_19_14:$LD_LIBRARY_PATH
      ```

      > **Note:**
      >
      > The Linux Instant Client download files are available as `.zip` files or `.rpm` files. You can use either version.

2. If you have not already done so, enable TLS connections on your Autonomous Database instance.

   See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for details.

3. Run Your Application

   a. Update your application to connect using your database username, your password, and the Oracle Net connect name given in the unzipped `tnsnames.ora` file. For example, user, `adb_user`, *password*, and `db2022adb_low` as the connect string.

   b. Alternatively, change the connect string in `tnsnames.ora` to match the string used by your application.

   c. Run your application.

Allowing TLS connections to Autonomous Database does not disallow mutual TLS (mTLS) connections. Both Mutual TLS (mTLS) and TLS connections are valid when an Autonomous Database instance is configured to allow TLS connections. See Connect Node.js and other Scripting Languages (mTLS) for information on connecting using mutual TLS (mTLS) with a wallet.

In this case, update your `sqlnet.ora` file by adding the following:

```
WALLET_LOCATION = (SOURCE = (METHOD = file) (METHOD_DATA = (DIRECTORY="/home/
wallet1")))
```

Prepare for JDBC Thin Connections

Applications that use JDBC Thin connections include the software necessary to make an Oracle Net Services connection. It is not necessary to download and install Oracle Client software.

Some applications use the JDK installed on your computer while others use a JDK that is embedded in the application installation. If your application uses the JDK installed on your computer and that JDK is version 8, 8u161 or later, no additional preparation is required. If your computer does not have JDK version 8, 8u161 or later, already installed then install the latest JDK first. You can download JDK version 8 from https://www.java.com/.

If your application is using a JDK version 8, prior to 8u161, then the JCE Policy Files must be updated within your application.

See Connect with JDBC Thin Driver for the steps required to use JDBC Thin connections to connect to an Oracle Database server.

- Set JVM Networking Properties
  Autonomous Database uses DNS names that map to multiple IP addresses (multiple load balancers) for better availability and performance. Depending on your application, you may want to configure certain JVM networking properties.

Set JVM Networking Properties

Autonomous Database uses DNS names that map to multiple IP addresses (multiple load balancers) for better availability and performance. Depending on your application, you may want to configure certain JVM networking properties.

For the Java Virtual Machine (JVM) address cache, any address resolution attempt caches the result whether it was successful or not, so that subsequent identical requests do not have to access the naming service. The address cache properties allow you to tune how the cache

operates. In particular, the `networkaddress.cache.ttl` value specifies the number of seconds a successful name lookup is kept in the cache. A value of -1, the default value, indicates a “cache forever” policy, while a value of `0` (zero) means no caching.

If your Java Virtual Machine (JVM) is configured to cache DNS address lookups, your application may be using only one IP address to connect to your Autonomous Database, resulting in lower throughput. To prevent this you can change your JVM's `networkaddress.cache.ttl` value to `0`, so that every connection request does a new DNS lookup. This ensures that different threads in your application are distributed over multiple load balancers.

To change the `networkaddress.cache.ttl` value for all applications, or in your application, do one of the following:

- Configure the security policy to set the value for all applications:
  Set `networkaddress.cache.ttl=0` in the file `$JAVA_HOME/jre/lib/security/java.security`
- Set the following property in your application code:

```
java.security.Security.setProperty("networkaddress.cache.ttl" , "0");
```

Connect Microsoft .NET, Visual Studio Code, and Visual Studio with a Wallet (mTLS)

Oracle Autonomous Database supports connectivity to the Microsoft .NET Framework, .NET Core, Visual Studio, and Visual Studio Code.

Oracle Data Provider for .NET (ODP.NET) provides run-time ADO.NET data access to a database.

ODP.NET has the following driver types:

- Unmanaged ODP.NET for .NET Framework applications
- Managed ODP.NET for .NET Framework applications
- ODP.NET Core for .NET Core applications

The following minimum versions are required:

- **ODP.NET**: 12.1 (Starting with April 2022 WINDBBP; ODP.NET Managed only), 18 (base release or later), 19.4 (or later, except for 19.10), or 21 (base release or later).

Oracle Developer Tools for Visual Studio and Oracle Developer Tools for VS Code provide database application design-time support in the Microsoft development environment, including tools for managing Autonomous Databases in the Oracle Cloud.

Oracle Developer Tools for VS Code provides database application design-time support in Visual Studio Code.

These software components are available as a free download from the following sites:

- Managed ODP.NET and ODP.NET Core: NuGet Gallery
- Unmanaged ODP.NET: Oracle Data Access Components Downloads
- Oracle Developer Tools for Visual Studio Code: VS Code Marketplace
- Oracle Developer Tools for Visual Studio: Oracle Developer Tools for Visual Studio

Oracle recommends using the latest provider and tools version with Oracle Autonomous Database.

Click the following link for instructions to download, install, and configure these components for use with mutual TLS (mTLS) and wallets:

- Developing .NET Applications for Oracle Autonomous Database

To use mutual TLS (mTLS) authentication to connect to Autonomous Database, you must download a wallet. See Download Client Credentials (Wallets) for information downloading wallets.

ODP.NET connectivity to Oracle Autonomous Database supports mutual TLS (mTLS) with wallets as well as TLS. See Connect Microsoft .NET, Visual Studio Code, and Visual Studio Without a Wallet for information on using TLS connections.

To learn more about using Oracle Autonomous Database and .NET, try the free .NET Development with Oracle Autonomous Database Quick Start. This lab walks you through setting up a .NET web server on Oracle Cloud Infrastructure that connects to Oracle Autonomous Database. Next, the lab guides developing and deploying a simple ASP.NET Core web application that uses all these components. By the end, you will have a live, working website on the Internet.

Connect Microsoft .NET, Visual Studio Code, and Visual Studio Without a Wallet

Oracle Autonomous Database supports connectivity to the Microsoft .NET Framework, .NET Core, Visual Studio, and Visual Studio Code using TLS authentication without a wallet.

Oracle Data Provider for .NET (ODP.NET) provides run-time ADO.NET data access to Autonomous Database. ODP.NET has the following driver types:

- Unmanaged ODP.NET for .NET Framework applications
- Managed ODP.NET for .NET Framework applications
- ODP.NET Core for .NET Core applications

Oracle Developer Tools for Visual Studio and Oracle Developer Tools for VS Code provide database application design-time support in the Microsoft development environment, including tools for managing Autonomous Databases in the Oracle Cloud.

Oracle Developer Tools for VS Code provides database application design-time support in Visual Studio Code.

These software components are available as a free download from the following sites:

- Managed ODP.NET and ODP.NET Core: NuGet Gallery
- Unmanaged ODP.NET: Oracle Data Access Components Downloads
- Oracle Developer Tools for Visual Studio Code: VS Code Marketplace
- Oracle Developer Tools for Visual Studio: Oracle Developer Tools for Visual Studio

Oracle recommends using the latest provider and tools version with Oracle Autonomous Database.

When you connect using TLS authentication with managed ODP.NET and ODP.NET Core you do not need to deploy the Oracle wallet or the Oracle network configuration files `sqlnet.ora` or `tnsnames.ora` with your application. Instead, you supply the data source attribute, a TLS connection string, with the configuration information in the ODP.NET connection.

To use TLS connections with Managed ODP.NET and ODP.NET Core, do the following:

1. Obtain managed ODP.NET or ODP.NET Core versions 19.13 or 21.4 (or above). Lower level versions do not support TLS connections with Oracle Autonomous Database.

2. Enable TLS connections on your Autonomous Database instance. See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for details.
3. After you enable TLS connections, supply a TLS connection string in the ODP.NET data source to connect to an Autonomous Database instance. See View TNS Names and Connection Strings for an Autonomous Database Instance for details on viewing or copying TLS connection strings.

   Notes for the TLS connection string:

   - The TLS connection string uses quotation marks around the distinguished name. If you store the TLS connection string in a .NET string, add a backslash escape sequence before each quotation mark (for example, \"). This allows .NET to recognize the quotation mark as part of the TLS connection string.
   - Verify that the connection string includes `(SECURITY=(SSL_SERVER_DN_MATCH=TRUE))` to ensure that the client matches the server DN. If not specified, add this to the connect string. For example:

     ```
     (description=
     (retry_count=20)(retry_delay=3)(address=(protocol=tcps)(port=1521)
     (host=HOSTNAME))(connect_data=(service_name=SERVICE_NAME))
     (security=(ssl_server_dn_match=true)))
     ```

Allowing TLS connections to Autonomous Database does not disallow mutual TLS (mTLS) connections. Both Mutual TLS (mTLS) and TLS connections are valid when an Autonomous Database instance is configured to allow TLS connections. See Connect Microsoft .NET, Visual Studio Code, and Visual Studio with a Wallet (mTLS) for information on connecting using mutual TLS (mTLS) with a wallet.

To learn more about using Oracle Autonomous Database and .NET, try the free .NET Development with Oracle Autonomous Database Quick Start. This lab walks you through setting up a .NET web server on Oracle Cloud Infrastructure that connects to Oracle Autonomous Database. Next, the lab guides developing and deploying a simple ASP.NET Core web application that uses all these components. By the end, you will have a live, working website on the Internet.

Connect Node.js and other Scripting Languages (mTLS)

You can use programs in different languages, including Python, Node.js, PHP, Ruby, R, Go, and Perl to connect to an Autonomous Database instance using mTLS (with wallets). Security is enforced using client credentials.

These scripting languages have database access APIs or drivers that use the Oracle Call Interface libraries.

> **Note:**
>
> Oracle Call Interface (OCI) clients support mTLS authentication with a wallet if you are connecting using the following client versions:
>
> - Oracle Instant Client/Oracle Database Client: 18.19 (or later), 19.2 (or later), or 21 (base release or later).
>
> For details on connecting Node.js or other scripting languages without a wallet, see Connect Node.js, and Other Scripting Languages Without a Wallet.

For additional details, see the following video:

- Connect Node.js Apps to Autonomous Transaction Processing

Install the Language Driver and Client Libraries

To connect to Autonomous Database from your scripting language, first install the language driver and client libraries as follows:

1. Install Oracle Instant Client/Oracle Database Client: 18.19 (or later), 19.2 (or later), or 21 (base release or later).

   The Instant Client works well for most applications. To install the Instant Client do the following:

   a. Select your desired architecture from the Instant Client Downloads page and download a Basic Package (available on the download page): Oracle Instant Client Downloads.

      Alternatively download the Basic Light Package from the download page for your desired architecture if the Basic Light globalization limitations suit your use.

   b. If you are building a language API or driver from source code, you may also need to download the Instant Client SDK: Oracle Instant Client

   c. Unzip the base package you selected. For example unzip to `C:\instantclient_12_2` or `/home/myuser/instantclient_18_5`. If you also download the SDK, unzip it in the same directory.

   d. On Windows, add the path to the `PATH` variable in the "System variables" section of the Environment Variables pane (for example add `C:\instantclient_12_2`). On Windows 8 access the `PATH` variable setting area by navigating to Control Panel>System>Advanced System Settings>Environment Variables. If you have multiple versions of Oracle libraries installed make sure the new directory occurs first in the path.

   e. On non-Windows platforms, create a symbolic link if it does not exist. For example:

      ```
      cd /home/myuser/instantclient_18_5
      ln -s libclntsh.so.18.1 libclntsh.so
      ```

      If there is no other Oracle software on your system that will be impacted, add Instant Client to the runtime link path. For example:

      ```
      sudo sh -c "echo /home/myuser/instantclient_18_5 > /etc/ld.so.conf.d/
      oic.conf"
      sudo ldconfig
      ```

      Alternatively set the library path in each shell that runs your application. For example:

      ```
      export LD_LIBRARY_PATH=/home/myuser/instantclient_18_5:$LD_LIBRARY_PATH
      ```

      > **Note:**
      >
      > The Linux Instant Client download files are available as `.zip` files or `.rpm` files. You can use either version.

2. Install the relevant language driver for Oracle Database:

- **Node.js** : To install node-oracledb for Node.js,, use the instructions on the following page: Installing node-oracledb.
- **ROracle**: To install ROracle for R, use the instructions on the following page: ROracle
- **PHP**: To install PHP OCI8 for PHP, use the instructions on the following page: Configuring PHP with OCI8.

  Windows DLLs are available on http://php.net/downloads.php and are also available from PECL oci8.
- **PHP PDO_OCI**: To install PHP PDO_OCI for PHP, use the instructions on the following page: Oracle Functions (PDO_OCI).

  Windows DLLs are available on http://php.net/downloads.php included in PHP.
- **Ruby**: To install ruby-oci8 for Ruby, use the instructions on the following page: Install for Oracle Instant Client
- **DBD for Perl**: To install DBD::Oracle for Perl, set `ORACLE_HOME` and your library search path such as `LD_LIBRARY_PATH` or `PATH` to the Instant Client directory and use the instructions on the following page: Installing DBD-Oracle.
- **Python**: For instructions for connecting Python applications, see Connect Python Applications with a Wallet (mTLS)

Enable Oracle Network Connectivity and Obtain the Security Credentials (Oracle Wallet)

1. Obtain client security credentials to connect to Autonomous Database. You obtain a zip file containing client security credentials and network configuration settings required to access your database. You must protect this file and its contents to prevent unauthorized database access. Obtain the client security credentials file as follows:
   - ADMIN user: Click **Database connection**. See Download Client Credentials (Wallets).
   - Other user (non-administrator): Obtain the Oracle Wallet from the administrator for your Autonomous Database.
2. Extract the client credentials (wallet) files:
   a. Unzip the client credentials zip file.
   b. If you are using Instant Client, make a `network/admin` subdirectory hierarchy under the Instant Client directory if necessary. Then move the files to this subdirectory. For example depending on the architecture or your client system and where you installed Instant Client, the files should be in the directory:

      ```
      C:\instantclient_12_2\network\admin
      ```

      or

      ```
      /home/myuser/instantclient_18_5/network/admin
      ```

      or

      ```
      /usr/lib/oracle/18.5/client64/lib/network/admin
      ```

      - If you are using a full Oracle Client move the file to `$ORACLE_HOME/network/admin`.

c. Alternatively, put the unzipped wallet files in a secure directory and set the `TNS_ADMIN` environment variable to that directory name.

> **Note:**
>
> From the zip file, only these files are required: `tnsnames.ora`, `sqlnet.ora`, `cwallet.sso`, and `ewallet.p12`.

3. If you are behind a proxy follow the steps in “Connections with an HTTP Proxy”, in Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections with Wallets (mTLS).

Run Your Application

1. Update your application to connect using your database username, your password, and the Oracle Net connect name given in the unzipped `tnsnames.ora` file. For example, user, `adb_user`, *password*, and `db2022adb_low` as the connect string.
2. Alternatively, change the connect string in `tnsnames.ora` to match the string used by your application.
3. Run your application.

Connect Node.js, and Other Scripting Languages Without a Wallet

You can use programs in different languages, including Python, Node.js, PHP, Ruby, R, Go, and Perl to connect to an Autonomous Database instance using TLS authentication without a wallet.

These scripting languages have database access APIs or drivers that use the Oracle Call Interface libraries. The Oracle Call Interface libraries can be either from the full Oracle Client or from Oracle Instant Client.

> **Note:**
>
> Oracle Call Interface (OCI) clients support TLS authentication without a wallet if you are using the following client versions:
>
> - Oracle Instant Client/Oracle Database Client 19.13 - only on Linux x64
> - Oracle Instant Client/Oracle Database Client 19.14 (or later) and 21.5 (or later) - only on Linux x64 and Windows

1. Install Oracle Instant Client.
   a. Got to the Oracle Instant Client page and click **Download Now**: Oracle Instant Client
   b. On the Oracle Instant Client Downloads page, select your platform.

      For example, under **Instant Client for Linux**, select the Instant Client for Linux x86-64 architecture (for this example, to download the Linux x86-64 version).
   c. Under **Version 19.14.0.0.0 (Requires glibc 2.14)**, select an Instant Client package to download.
   d. If you are building a language API or driver from source code, you may also need to download the Instant Client SDK Package version 19.14: Oracle Instant Client

e. Unzip the base package you selected. If you also download the SDK, unzip it in the same directory.

f. On Linux, create a symbolic link if it does not exist. For example:

```
cd /home/myuser/instantclient_19_14
ln -s libclntsh.so.19.1 libclntsh.so
```

If there is no other Oracle software on your system that will be impacted, add Instant Client to the runtime link path. For example:

```
sudo sh -c "echo /home/myuser/instantclient_19_14 > /etc/ld.so.conf.d/
oic.conf"
sudo ldconfig
```

Alternatively set the library path in each shell that runs your application. For example:

```
export LD_LIBRARY_PATH=/home/myuser/instantclient_19_14:$LD_LIBRARY_PATH
```

> **Note:**
>
> The Linux Instant Client download files are available as `.zip` files or `.rpm` files. You can use either version.

2. Install the relevant language driver for Oracle Database:

- **Node.js** : To install node-oracledb for Node.js,, use the instructions on the following page: Installing node-oracledb.
- **ROracle**: To install ROracle for R, use the instructions on the following page: ROracle
- **PHP**: To install PHP OCI8 for PHP, use the instructions on the following page: Configuring PHP with OCI8.

  Windows DLLs are available on http://php.net/downloads.php and are also available from PECL oci8.
- **PHP PDO_OCI**: To install PHP PDO_OCI for PHP, use the instructions on the following page: Oracle Functions (PDO_OCI).

  Windows DLLs are available on http://php.net/downloads.php included in PHP.
- **Ruby**: To install ruby-oci8 for Ruby, use the instructions on the following page: Install for Oracle Instant Client
- **DBD for Perl**: To install DBD::Oracle for Perl, set `ORACLE_HOME` and your library search path such as `LD_LIBRARY_PATH` or `PATH` to the Instant Client directory and use the instructions on the following page: Installing DBD-Oracle.
- **Python**: For instructions for connecting Python applications, see Connect Python Applications Without a Wallet (TLS)

3. If you have not already done so, enable TLS connections on your Autonomous Database instance.

   See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for details.

4. Run Your Application

a. Update your application to connect using your database username, your password, and the Oracle Net connect name given in the unzipped `tnsnames.ora` file. For example, user, `adb_user`, *password*, and `db2022adb_low` as the connect string.

b. Alternatively, change the connect string in `tnsnames.ora` to match the string used by your application.

c. Run your application.

Allowing TLS connections to Autonomous Database does not disallow mutual TLS (mTLS) connections. Both Mutual TLS (mTLS) and TLS connections are valid when an Autonomous Database instance is configured to allow TLS connections. See Connect Node.js and other Scripting Languages (mTLS) for information on connecting using mutual TLS (mTLS) with a wallet.

Connect Power BI and Microsoft Data Tools to Autonomous Database

Oracle Client for Microsoft Tools (OCMT) is a graphical user interface (GUI) native Microsoft Software installer (MSI) that simplifies ODP.NET setup and Oracle database connectivity to multiple Microsoft data tools.

OCMT supports the following Microsoft tools:

- Power BI Desktop (unmanaged ODP.NET)
- Power BI service (unmanaged ODP.NET)
- Excel (unmanaged ODP.NET)
- SQL Server Analysis Services (unmanaged ODP.NET)
- BizTalk Server (unmanaged ODP.NET)
- SQL Server Data Tools (managed ODP.NET)
- SQL Server Integration Services (managed ODP.NET)
- SQL Server Reporting Services (managed ODP.NET)

To begin with, you must download this tool from the Oracle Client for Microsoft Tools page by clicking the **64-bit Oracle Client for Microsoft Tools** link. Save the **Oracle-Client-for-Microsoft-Tools.exe** file to your Windows machine and double-click the file to run the GUI installer.

By default, OCMT installs with both the managed and unmanaged ODP.NET providers. However, you can choose to have only managed ODP.NET or unmanaged ODP.NET driver while installing the Oracle client. The ODP.NET driver type each Microsoft data tool uses is shown in parenthesis in the above list.

Managed ODP.NET works with both 64-bit and 32-bit MS tool runtimes. In the case of unmanaged ODP.NET provider, OCMT installs the 64-bit driver. Thus, it supports 64-bit tools, such as 64-bit Power BI Desktop and 64-bit Excel, but not their 32-bit counterparts. 32-bit MS tools can continue using 32-bit Oracle Data Access Client (ODAC) 19c xcopy installs to set up 32-bit unmanaged ODP.NET as before.

Irrespective of the Microsoft tool that you want to connect to your Autonomous Database, you must follow a sequence of steps to use OCMT. They are:

- Provision an Autonomous Database. Refer to Provision Autonomous Database for instructions.
- Download database credentials to the Windows client. Refer to Download Client Credentials (Wallets) for instructions.

- Depending on the Microsoft (MS) tool that you want to connect to the Autonomous Database, you can install any of the following MS tools to the Windows client:
  - Visual Studio and Microsoft Reporting Services extension
  - Visual Studio and Microsoft Analysis Services extension
  - Visual Studio with the SQL Server Integration Services extension, which also supports SQL Server Data Tools
  - Power BI On-premises Data gateway
  - Power BI Desktop
- Install and configure ODP.NET on the Windows client
- Validate the MS tool connection to Autonomous Database

Below are step-by-step connectivity tutorials for each of the MS tools listed above. These tutorials show how to connect these MS tools with either on-premises Oracle databases or Oracle Autonomous Database:

- Power BI Desktop: Connect to Oracle Database
- Power BI Service: Connect to Oracle Database
- SQL Server Data Tools and Integration Services: Connect to Oracle Database
- SQL Server Analysis Services: Connect to Oracle Database
- SQL Server Reporting Services: Connect to Oracle Database

Connect Python Applications to Autonomous Database

You can connect Python applications to your Autonomous Database instance either with a wallet (mTLS) or without a wallet (TLS).

- Install Python and the python-oracledb Driver
  To connect to Autonomous Database from your Python application, install Python and the python-oracledb driver.
- Connect Python Applications Without a Wallet (TLS)
- Connect Python Applications with a Wallet (mTLS)

Install Python and the python-oracledb Driver

To connect to Autonomous Database from your Python application, install Python and the python-oracledb driver.

1. Install Python 3, if it is not already available.

   The version of Python you use depends on your client-side OS and hardware. For example Windows, Linux, macOS, and others.

   > **Note:**
   >
   > Oracle recommends you keep up to date with Python and python-oracledb driver releases.

2. Install the python-oracledb driver from PyPI.

The python-oracledb driver is a Python programming language extension module allowing Python programs to connect to Oracle Database. The python-oracledb driver is the renamed, new major release of the popular cx_Oracle driver.

Supported python-oracledb driver versions: python-oracledb 1.0 (or later)

Run the following command to upgrade python:

```
python -m pip install oracledb --upgrade
```

You should see output similar to the following:

```
Collecting oracledb
  Downloading oracledb-1.0.3-cp310-cp310-win_amd64.whl (1.0 MB)
     ---------------------------------------- 1.0/1.0 MB 1.8 MB/s eta
0:00:00
Collecting cryptography>=3.4
  Downloading cryptography-37.0.4-cp36-abi3-win_amd64.whl (2.4 MB)
     ---------------------------------------- 2.4/2.4 MB 3.5 MB/s eta
0:00:00
Collecting cffi>=1.12
  Downloading cffi-1.15.1-cp310-cp310-win_amd64.whl (179 kB)
     ---------------------------------------- 179.1/179.1 kB 5.4 MB/s eta
0:00:00
Collecting pycparser
  Downloading pycparser-2.21-py2.py3-none-any.whl (118 kB)
     ---------------------------------------- 118.7/118.7 kB 7.2 MB/s eta
0:00:00
Installing collected packages: pycparser, cffi, cryptography, oracledb
Successfully installed cffi-1.15.1 cryptography-37.0.4 oracledb-1.0.3
pycparser-2.21
```

Notes for installing python-oracledb:

- If you are behind a proxy, use the `--proxy` option to add a proxy server to the command. For example:

  ```
  python -m pip install oracledb --upgrade --proxy=http://
  proxy.example.com:80
  ```

- In the case where you do not have permission to write to system directories, include the `--user` option. For example:

  ```
  python -m pip install oracledb --upgrade --user
  ```

- If a binary package is not available for your platform, running `pip` will download the source package instead. The source is compiled and the resulting binary is installed.

See Installing python-oracledb for additional options and tips.

3. If you want to use the python-oracledb driver in Thick mode, install Oracle Client software.

   By default, python-oracledb runs in Thin mode which connects directly to Oracle Database. Thin mode does not require Oracle Client libraries. However, some additional functionality is available when python-oracledb runs in Thick mode.

> **Note:**
>
> See Oracle Database Features Supported by python-oracledb for information on supported features in python-oracledb Thin and Thick modes. Not all of the features shown in this link are available with Autonomous Database.

Python-oracledb uses Thick mode when you use either the Oracle Instant client libraries or the Oracle Database Client libraries and you call `oracledb.init_oracle_client()` in your Python code.

When you install Oracle Client Software, there are differences in required minimum versions for mTLS and TLS connections, as follows:

- **Mutual TLS (mTLS) Connections**:
  - If your database is on a remote computer, then download the free Oracle Instant Client “Basic” or “Basic Light” package for your operating system architecture. Use a supported version: **Oracle Instant Client**: 18.19 (or later), 19.2 (or later), or 21 (base release or later).
  - Alternatively, you can use the Full Oracle Database client libraries when they are available on your system (including Full Oracle Database Client: **Oracle Database Client**: 18.19 (or later), 19.2 (or later), or 21 (base release or later).
- **TLS Connections**: Oracle Call Interface (OCI) clients support TLS authentication without a wallet if you are using the following client versions:
  - Oracle Instant Client/Oracle Database Client 19.14 (or later) and 21.5 (or later) - only on Linux x64 and Windows
  - Alternatively, you can use the Full Oracle Database client libraries when they are available on your system, including Full Oracle Database Client 19.14 (or later) and 21.5 (or later).

Connect Python Applications Without a Wallet (TLS)

You can connect Python applications to your Autonomous Database instance without a wallet. Connecting a Python application without a wallet (TLS) provides security for authentication and encryption, and security is enforced using client credentials (by providing a username and password).
The python-oracledb driver's default "Thin mode" connects directly to Oracle Database. It can optionally use Oracle Client libraries, "Thick mode", for some additional functionality. The Oracle Client libraries can be from Oracle Instant Client, the full Oracle Client, or an Oracle Database installation.

Follow these steps to connect your Python application to an Autonomous Database instance without a wallet (TLS):

1. Install Python and the python-oracledb Driver
2. Enable TLS on Autonomous Database and Obtain Connection String
3. Run Python Application Without a Wallet

See Enabling python-oracledb Thick mode for information on Thick mode.

Topics

- Enable TLS on Autonomous Database and Obtain Connection String

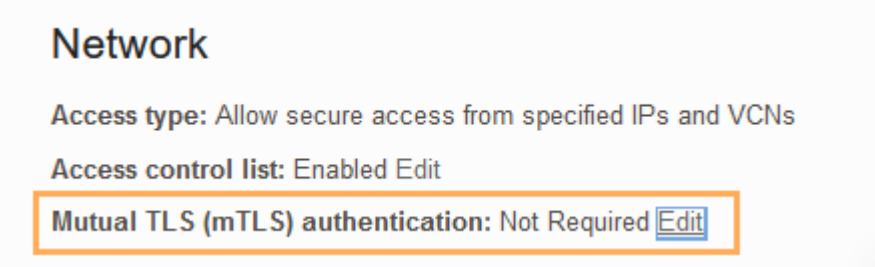
- Run Python Application Without a Wallet
  A Python application can connect to your Autonomous Database instance without a wallet (TLS) using the database credentials and a connect descriptor.

Enable TLS on Autonomous Database and Obtain Connection String

To run a Python application without a wallet, enable the Autonomous Database instance for TLS connections and obtain a connection string to contact the database from the Python application.

1. Determine if your Autonomous Database instance is enabled for TLS connections.

   If the instance is enabled for TLS connections, in the **Network** area on the Oracle Cloud Infrastructure Console the **Mutual TLS (mTLS) authentication** field shows: **Not Required**:

   If your instance requires Mutual TLS authentication, allow TLS connections on your Autonomous Database instance. See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for details.

2. Obtain an Autonomous Database service connection string to access the database as follows:

   a. On the Oracle Cloud Infrastructure Console, click **Database connection**.

   b. Select **TLS** in the Database Connection dialog box, under **Connection Strings**, in the **TLS Authentication** drop-down list.

   > **Note:**
   >
   > You must select **TLS** in the **TLS Authentication** drop-down to obtain the TLS connection strings before you copy a connection string (when the value is **Mutual TLS** the connection strings have different values and do not work with TLS connections).

   c. Copy the Connection String for the database service you want to use with your application.

   See View TNS Names and Connection Strings for an Autonomous Database Instance for more information.

Run Python Application Without a Wallet

A Python application can connect to your Autonomous Database instance without a wallet (TLS) using the database credentials and a connect descriptor.

1. Obtain the connection string, as described in Enable TLS on Autonomous Database and Obtain Connection String.

2. In your Python application, set the following connection parameters to connect to an Autonomous Database instance:

- `dsn`: Use the connection string to specify the desired database service name.
- `password`: Specifies the database user password.
- `user`: Specifies the database user.

For example:

```
cs='''(description = (retry_count=20)(retry_delay=3)
(address=(protocol=tcps)
     (port=1522)(host=xxx.oraclecloud.com))
     (connect_data=(service_name=xxx.adb.oraclecloud.com))
     (security=(ssl_server_dn_match=yes)))'''

connection=oracledb.connect(
     user="admin",
     password=password,
     dsn=cs)
```

3. If you want to connect in Thick mode, include `oracledb.init_oracle_client()` in your Python application.

   For example:

```
cs='''(description = (retry_count=20)(retry_delay=3)
(address=(protocol=tcps)
     (port=1522)(host=xxx.oraclecloud.com))
     (connect_data=(service_name=xxx.adb.oraclecloud.com))
     (security=(ssl_server_dn_match=yes)))'''

oracledb.init_oracle_client()
connection=oracledb.connect(
     user="admin",
     password=password,
     dsn=cs)
```

Connect Python Applications with a Wallet (mTLS)

You can connect Python applications to your Autonomous Database instance with a wallet.

Connecting a Python application with a wallet (mTLS) provides enhanced security for authentication and encryption, and security is enforced using client credentials (by providing a username and password).

The python-oracledb driver's default "Thin mode" connects directly to Oracle Database. The driver can optionally use Oracle Client libraries, "Thick mode", for some additional functionality. The Oracle Client libraries can be from Oracle Instant Client, the full Oracle Client, or from an Oracle Database installation.

Follow these steps to connect your Python application to an Autonomous Database instance using a wallet (mTLS):

1. Install Python and the python-oracledb Driver
2. Obtain Security Credentials (Oracle Wallet) and Enable Network Connectivity
3. Perform this step if you only want to connect in Thin mode: Run Python Application with python-oracledb Thin Mode with a Wallet (mTLS)

4. Perform this step if you want to connect in Thick mode: Run Python Application with python-oracledb Thick Mode with a Wallet (mTLS)

Topics

- Obtain Security Credentials (Oracle Wallet) and Enable Network Connectivity
  Obtain client security credentials to connect to an Autonomous Database instance.
- Run Python Application with python-oracledb Thin Mode with a Wallet (mTLS)
  By default, python-oracledb uses Thin mode to connect directly to your Autonomous Database instance.
- Run Python Application with python-oracledb Thick Mode with a Wallet (mTLS)

Obtain Security Credentials (Oracle Wallet) and Enable Network Connectivity

Obtain client security credentials to connect to an Autonomous Database instance.

1. Download a wallet file from the Autonomous Database instance to obtain a zip file that contains the client security credentials and network configuration settings required to access an Autonomous Database instance.

   Obtain the client security credentials (`wallet.zip` file):

   - ADMIN user: On the Oracle Cloud Infrastructure Console, click **Database connection**. See Download Client Credentials (Wallets).
   - Other user (non-administrator): Obtain the Oracle Wallet from the administrator for your Autonomous Database instance.

   > **Note:**
   >
   > Protect the `wallet.zip` file and its contents to prevent unauthorized database access.

2. Unzip the client credentials file (`wallet.zip`).

Run Python Application with python-oracledb Thin Mode with a Wallet (mTLS)

By default, python-oracledb uses Thin mode to connect directly to your Autonomous Database instance.

In Thin mode only two files from the wallet zip are needed:

- `tnsnames.ora`: Maps net service names used for application connection strings to your database services.
- `ewallet.pem`: Enables SSL/TLS connections in Thin mode.

To connect in Thin mode:

1. Move `tnsnames.ora` and `ewallet.pem` files to a location on your system.

   - Linux
   - Windows

Linux

For example on Linux:

```
/opt/OracleCloud/MYDB
```

Windows

For example on Windows:

```
C:\opt\OracleCloud\MYDB
```

2. In your Python application, set the following connection parameters to connect to an Autonomous Database instance:
   - `config_dir`: Specifies the directory containing `tnsnames.ora`.
   - `dsn`: Use to specify the desired network alias from the `tnsnames.ora` file.
   - `password`: Specifies the database user password.
   - `user`: Specifies the database user.
   - `wallet_location`: Specifies the directory containing the PEM file (`ewallet.pem`).
   - `wallet_password`: Specifies the password for the PEM file (`ewallet.pem`). You set this password when you download the `wallet.zip` file.

- Linux
- Windows

Linux

For example, on Linux to connect as the ADMIN user using `oracledb.connect` with the `db2024_low` network service name (the service name is found in `tnsnames.ora`):

```
connection=oracledb.connect(
     config_dir="/opt/OracleCloud/MYDB",
     user="admin",
     password=password,
     dsn="db2024_low",
     wallet_location="/opt/OracleCloud/MYDB",
     wallet_password=wallet_pw)
```

Windows

For example, on Windows to connect as the ADMIN user using `oracledb.connect` with the `db2024_low` network service name (the service name is found in `tnsnames.ora`):

```
connection=oracledb.connect(
     config_dir=r"C:\opt\OracleCloud\MYDB",
     user="admin",
     password=password,
```

```
    dsn="db2024_low",
    wallet_location=r"C:\opt\OracleCloud\MYDB",
    wallet_password=wallet_pw)
```

The use of a 'raw' string `r"..."` means that backslashes are treated as directory separators.

As shown in this example, `wallet_location` and `config_dir` are set to the same directory (and the directory contains `tnsnames.ora` and `ewallet.pem`). Specifying the same directory for these files is not required.

If you are behind a firewall, you can tunnel TLS/SSL connections through a proxy using HTTPS_PROXY in the connect descriptor or by setting connection attributes. Successful connection depends on specific proxy configurations. Oracle does not recommend using a proxy in a production environment, due to the possible impact on performance.

In Thin mode you can specify a proxy by adding the `https_proxy` and `http_proxy_port` parameters.

For example, on Linux:

```
connection=oracledb.connect(
     config_dir="/opt/OracleCloud/MYDB",
     user="admin",
     password=password,
     dsn="db2024_low",
     wallet_location="/opt/OracleCloud/MYDB",
     wallet_password=wallet_pw,
     https_proxy='myproxy.example.com',
     https_proxy_port=80)
```

For example, on Windows:

```
connection=oracledb.connect(
     config_dir=r"C:\opt\OracleCloud\MYDB",
     user="admin",
     password=password,
     dsn="db2024_low",
     wallet_location=r"C:\opt\OracleCloud\MYDB",
     wallet_password=wallet_pw,
     https_proxy='myproxy.example.com',
     https_proxy_port=80)
```

Run Python Application with python-oracledb Thick Mode with a Wallet (mTLS)

By default, python-oracledb runs in Thin mode which connects directly to Oracle Database. Additional python-oracledb features are available when the driver runs in Thick mode.

> **Note:**
>
> Thick mode requires that the Oracle Client libraries are installed where you run Python. You must also call `oracledb.init_oracle_client()` in your Python code.

In Thick mode the following three files from the wallet zip file are required:

- `tnsnames.ora`: Contains the net service names used for application connection strings and maps the strings to your database services.
- `sqlnet.ora`: Specifies the SQL*Net client side configuration.
- `cwallet.sso`: Contains the auto-open SSO wallet.

To connect in Thick mode:

1. Place the files `tnsnames.ora`, `sqlnet.ora`, and `cwallet.sso` on your system.

   Use one of two options to place these files on your system:

   - If you are using Instant Client, move the files to a `network/admin` subdirectory hierarchy under the Instant Client directory. For example depending on the architecture or your client system and where you installed Instant Client, the files should placed be in a directory location such as:

     ```
     /home/myuser/instantclient_19_21/network/admin
     ```

     or

     ```
     /usr/lib/oracle/19.21/client64/lib/network/admin
     ```

     For example, on Linux if you are using the full Oracle Client move the files to `$ORACLE_HOME/network/admin`.

   - Alternatively, move the files to any accessible directory.

     For example, on Linux move the files to the directory `/opt/OracleCloud/MYDB` and edit `sqlnet.ora` to change the wallet location directory to the directory containing the `cwallet.sso` file.

     For example, on Linux edit `sqlnet.ora` as follows:

     ```
     WALLET_LOCATION = (SOURCE = (METHOD=file) (METHOD_DATA =
     (DIRECTORY="/opt/OracleCloud/MYDB")))
     SSL_SERVER_DN_MATCH=yes
     ```

     When the configuration files are not in the default location, your application needs to indicate where they are, either with the `config_dir` parameter in the call `oracledb.init_oracle_client()` or by setting the `TNS_ADMIN` environment variable.

     > **Note:**
     >
     > Neither of these settings are needed, and you do not need to edit `sqlnet.ora` if you put all the configuration files in the `network/admin` directory.

2. In your Python application set the following initialization and connection parameters to connect to the Autonomous Database instance:

   - `config_dir`: Specifies the configuration directory when you are putting the configuration files. This is only required when the configuration files are placed in a directory outside of the instant client configuration directory `network/admin`.

- `dsn`: Specifies the desired network alias from the `tnsnames.ora` file.
- `password`: Specifies the database user password.
- `user`: Specifies the database user.

In the first case for placement of the configuration files, connect to the Autonomous Database instance using your database credentials by setting the `dsn` parameter to the desired network alias from `tnsnames.ora`.

For example, to connect as the ADMIN user using `oracledb.init_oracle_client` and connect with the `db2024_low` network service name (where the service name is found in `tnsnames.ora`):

```
oracledb.init_oracle_client()
   connection=oracledb.connect(
       user="admin",
       password=password,
       dsn="db2024_low")
```

When configuration files are in a directory outside of the instant client configuration directory, set the `config_dir` parameter when you call `oracledb.init_oracle_client`.

- Linux
- Windows

Linux

For example, on Linux to connect as the ADMIN user using the `db2024_low` network service name:

```
oracledb.init_oracle_client(config_dir="/opt/OracleCloud/MYDB")
   connection=oracledb.connect(
      user="admin",
      password=password,
      dsn="db2024_low")
```

Windows

For example, on Windows to connect as the ADMIN user using the `db2024_low` network service name:

```
oracledb.init_oracle_client(config_dir=r"C:\opt\OracleCloud\MYDB")
   connection=oracledb.connect(
      user="admin",
      password=password,
      dsn="db2024_low")
```

The use of a ‘raw’ string `r"..."` means that backslashes are treated as directory separators.

If you are behind a firewall, you can tunnel TLS/SSL connections through a proxy using HTTPS_PROXY in the connect descriptor or by setting connection attributes. Successful connection depends on specific proxy configurations. Oracle does not recommend using a proxy in a production environment, due to the possible impact on performance.

In Thick mode you can specify a proxy by editing the `sqlnet.ora` file and adding a line:

```
SQLNET.USE_HTTPS_PROXY=on
```

In addition, edit `tnsnames.ora` and add an `HTTPS_PROXY` proxy name and `HTTPS_PROXY_PORT` port to the connect descriptor address list of any service name you plan to use.

For example:

```
mydb_high=(description=
(address=(https_proxy=myproxy.example.com)
(https_proxy_port=80)
(protocol=tcps)(port=1522)(host=...)
```

See Enabling python-oracledb Thick mode for information on Thick mode.

Download Database Connection Information

You can download Oracle client credentials, wallet files, from an Autonomous Database instance.

If you are not an Autonomous Database administrator and your application requires a wallet to connect, then your administrator should provide you with the client credentials. You can also view TNS names and connection strings for your database.

- Download Client Credentials (Wallets)
  To download client credentials you can use the Oracle Cloud Infrastructure Console or Database Actions.
- Wallet README File
  The wallet `README` file contains the wallet expiration information and details for Autonomous Database tools and resources.
- View TNS Names and Connection Strings for an Autonomous Database Instance
  From the Database Connection page on the Oracle Cloud Infrastructure Console you can view Autonomous Database TNS names and connection strings.

Download Client Credentials (Wallets)

To download client credentials you can use the Oracle Cloud Infrastructure Console or Database Actions.

> **Note:**
>
> The password you provide when you download the wallet protects the downloaded Client Credentials wallet.

For commercial regions, the wallet password complexity for the password you supply requires the following:

- Minimum of 8 characters
- Minimum of 1 letter
- Minimum of 1 numeric character or 1 special character

For US Government regions, the wallet password complexity requires all of the following:

- Minimum of 15 characters
- Minimum of 1 lowercase letter
- Minimum of 1 uppercase letter
- Minimum of 1 numeric character
- Minimum 1 special character

To download client credentials from the Oracle Cloud Infrastructure Console:

1. Navigate to the Autonomous Database details page.
2. Click **Database connection**.
3. On the **Database connection** page select the **Wallet type**:
   - **Instance wallet**: Wallet for a single database only; this provides a database-specific wallet.
   - **Regional wallet**: Wallet for all Autonomous Databases for a given tenant and region (this includes all service instances that a cloud account owns).

   > **Note:**
   >
   > Oracle recommends you provide a database-specific wallet, using **Instance wallet**, to end users and for application use whenever possible. Regional wallets should only be used for administrative purposes that require potential access to all Autonomous Databases within a region.

4. Click **Download wallet**.
5. In the **Download wallet** dialog, enter a wallet password in the **Password** field and confirm the password in the **Confirm password** field.
6. Click **Download** to save the client security credentials zip file.

   By default the filename is: `Wallet_`*databasename*`.zip`. You can save this file as any filename you want.

   You must protect this file to prevent unauthorized database access.

To download client credentials from Database Actions:

First, access Database Actions as the ADMIN user. See Access Database Actions as ADMIN for more information.

1. Access Database Actions as the ADMIN user. See Access Database Actions as ADMIN for more information.
2. On the Database Actions Launchpad, under **Administration**, select **Download Client Credentials (Wallet)**.
3. On the **Download Client Credentials (Wallet)** page, enter a wallet password in the **Password** field and confirm the password in the **Confirm Password** field.

4. Click **Download** to save the client security credentials zip file. By default the filename is: `Wallet_`*databasename*`.zip`. You can save this file as any filename you want. You must protect this file to prevent unauthorized database access.

> **Note:**
>
> When you use Database Actions to download a wallet there is no **Wallet type** option on the **Download Client Credentials (Wallet)** page and you always download an instance wallet. If you need to download the regional wallet click **Database connection** on the Oracle Cloud Infrastructure Console.

The zip file includes the following:

File	Description
`cwallet.sso`	Auto-open SSO wallet
`ewallet.p12`	PKCS12 file. The PKCS12 file is protected by the wallet password provided while downloading the wallet.
`ewallet.pem`	Encoded certificate file used to authenticate with certificate authority (CA) server certificate.
`keystore.jks`	Java keystore file. This file is protected by the wallet password provided while downloading the wallet.
`ojdbc.properties`	Contains the wallet related connection property required for JDBC connection. This should be in the same path as `tnsnames.ora`.
`README`	Contains wallet expiration information and links for Autonomous Database tools and resources. See Wallet README File for information on the contents of the `README` file.
`sqlnet.ora`	SQL*Net client side configuration.
`tnsnames.ora`	Network configuration file storing connect descriptors.
`truststore.jks`	Java truststore file. This file is protected by the wallet password provided while downloading the wallet.

Notes for wallet files and the wallet password:

- To invalidate database client certification keys associated with a wallet, see Rotate Wallets with Immediate Rotation.
- Wallet files, along with the Database user ID and password provide access to data in your database. Store wallet files in a secure location. Share wallet files only with authorized users. If wallet files are transmitted in a way that might be accessed by unauthorized users (for example, over public email), transmit the wallet password separately and securely.
- For better security, Oracle recommends using restricted permissions on wallet files. This means setting the file permissions to 600 on Linux/Unix. Similar restrictions can be achieved on Windows by letting the file owner have Read and Write permissions while all other users have no permissions.

- Autonomous Database uses strong password complexity rules for all users based on Oracle Cloud security standards. For more information on the password complexity rules see Create Users on Autonomous Database - Connecting with a Client Tool.
- The `README` file that contains wallet expiration information is not available in wallet zip files that were downloaded before April 2020.
- Starting six weeks before the wallet expiration date Autonomous Database sends notification emails each week, indicating the wallet expiration date. These emails provide notice before your wallet expires that you need to download a new wallet. You will receive these notification emails only if there is a connection that uses a wallet that is about to expire.

  You can also use the `WalletExpirationWarning` event to be notified when a wallet is due to expire. You will receive these notification events only if you are subscribed to Critical events and there is a connection that uses a wallet that is about to expire. See About Events Based Notification and Automation on Autonomous Database for more information.

Wallet README File

The wallet `README` file contains the wallet expiration information and details for Autonomous Database tools and resources.

The wallet expiration information at the top of the `README` file shows the following information:

- The date when the wallet was downloaded.
- The date when the wallet SSL certificate provided in the wallet expires. If your wallet is nearing expiration or is expired, then download a new wallet or obtain a new wallet from your Autonomous Database administrator. If you do not download a new wallet before the expiration date, you will no longer be able to connect to your database.

The Autonomous Database tools and resources area provides the following information:

Tool or Resource	Description
Database Actions	Load, explore, transform, model, and catalog your data. Use an SQL worksheet, build REST interfaces and low-code apps, manage users and connections, build and apply machine learning models. Access Link: provides the link to use Database Actions. See Connect with Built-In Oracle Database Actions for more information.
Graph Studio	Oracle Graph Studio lets you create scalable property graph databases. Graph Studio automates the creation of graph models and in-memory graphs from database tables. It includes notebooks and developer APIs that allow you to execute graph queries using PGQL (an SQL-like graph query language) and over 50 built-in graph algorithms. Graph Studio also offers dozens of visualization, including native graph visualization. Access Link provides the link to use Graph Studio. See About Oracle Graph Studio on Autonomous Database for more information.
Oracle APEX	Oracle APEX is a low-code development platform that enables you to build scalable, secure enterprise apps that can be deployed anywhere. Access Link: provides the link to use Oracle APEX. See Access Oracle APEX Administration Services for more information.

Tool or Resource	Description
Oracle Machine Learning User Management	Create new Oracle Machine Learning user accounts and manage the credentials for existing Oracle Machine Learning users. Access Link: provides the link to use Oracle Machine Learning User Management. See Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database for more information.
Oracle Machine Learning User Notebooks	Oracle Machine Learning notebooks provide easy access to Oracle's parallelized, scalable in-database implementations of a library of Oracle Advanced Analytics' machine learning algorithms (classification, regression, anomaly detection, clustering, associations, attribute importance, feature extraction, times series, and so on), SQL, PL/SQL and Oracle's statistical and analytical SQL functions. Access Link: provides the link to use Oracle Machine Learning User Notebooks. See Work with Oracle Machine Learning User Interface for Data Access, Analysis, and Discovery for more information.
SODA Drivers	Simple Oracle Document Access (SODA) is a set of APIs that let you work with JSON documents managed by the Oracle Database without needing to use SQL. SODA drivers are available for REST, Java, Node.js, Python, PL/SQL, and C. Access Link: provides the link to download the SODA drivers. See Work with Simple Oracle Document Access (SODA) in Autonomous Database for more information.

Notes for wallet `README` file:

- If you rename your Autonomous Database instance, the tools links change and the old links no longer work. To obtain valid tools links you must download a new Wallet zip file with an updated `README` file. The SODA drivers link is a resource link and this link does not change when you rename an instance.
- The `README` in a regional wallet does not contain the Autonomous Database tools and resources links.

View TNS Names and Connection Strings for an Autonomous Database Instance

From the Database Connection page on the Oracle Cloud Infrastructure Console you can view Autonomous Database TNS names and connection strings.

> **Note:**
>
> See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for information on allowing TLS connections.

Perform the following steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.

- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To view the TNS names and connection strings, do the following:

1. On the Autonomous Database details page, click **Database connection**.

   By default this shows the Mutual TLS connection information in a table with the TNS names and connection strings for the Autonomous Database instance.

2. When both Mutual TLS (mTLS) and TLS connections are allowed, under **TLS authentication** select **TLS** to view the TNS names and connection strings for connections with TLS authentication.

   The TNS names are the same for mTLS and TLS authentication. The connection strings differ for mTLS and TLS connections, with different port definitions. Mutual TLS (mTLS) connections use port 1522. TLS connections use port 1521.

   In the Connection String column, click **Show** to display the full value of a connection string or click **Copy** to copy a connection string.

   For example, when you click **Show** you see the full connection string.

Database connection

Help

Connections to your Autonomous Database are secured, and can be authorized using TLS or mTLS authentication options. TLS authentication is easier to use, provides better connection latency, and does not require you to download client credentials (wallet) if any of these is true for your connections:

- You are using JDBC Thin Client (version 12.2.0.1 or higher) with JDK 8(u163+) or higher.
- You are using the Python python-oracledb driver.
- You are using ODP.NET version 19.14 (or higher), or 21.5 (or higher).
- You are using an Oracle Call Interface based driver with Oracle Client libraries version 19.14 (or higher), or 21.5 (or higher).

Learn more about TLS authentication and how to enable it.

Download client credentials (Wallet)

To download your client credentials, select the wallet type, and click **Download wallet**. You then enter a password for the wallet. This client credential download only contains information for mTLS connections. **You do not need a wallet for TLS connections.**

Wallet type

Instance Wallet

Download wallet | Rotate wallet

Wallet last rotated: -

Connection Strings

Use the following connection strings or TNS names for your connections. See the documentation for details.

TLS Authentication

TLS

TNS Name	Connection String
fmexample1_high	(description= (retry_count=15)(retry_delay=3)(address=(protocol=tcps)(port=1521)(host=adb.us-ashburn-1.oraclecloud.com))(connect_data=(service_name=gvenbck_fmexample1_high.adb.oraclecloud.com))(security=(ssl_server_dn_match=yes))) Hide Copy
fmexample1_low	...ecurity=(ssl_server_dn_match=yes))) Show Copy
fmexample1_medium	...ecurity=(ssl_server_dn_match=yes))) Show Copy

Showing 3 Items

Close

Connect to Autonomous Database Using Oracle Database Tools

Oracle Database Tools such as SQL Developer, SQL*Plus, and SQLcl can be used with Autonomous Database.

The following sections provide step-by-step instructions for connecting to Autonomous Database using these tools.

- Connect Oracle SQL Developer with a Wallet (mTLS)
  Oracle SQL Developer is a free integrated development environment that simplifies the development and management of Autonomous Database.
- Connect Oracle SQL Developer Without a Wallet
  Oracle SQL Developer is a free integrated development environment that simplifies the development and management of Autonomous Database. Oracle SQL Developer provides support for connecting using TLS authentication without a wallet.
- Connect SQL*Plus with a Wallet (mTLS)
  SQL*Plus is a command-line interface used to enter SQL commands. SQL*Plus connects to an Oracle database.
- Connect SQL*Plus Without a Wallet
  SQL*Plus is a command-line interface used to enter SQL commands. SQL*Plus connects to an Oracle database.
- Connect Oracle SQLcl Cloud with a Wallet (mTLS)
  SQLcl is a command-line interface used to enter SQL commands. You can use SQLcl to connect to an Autonomous Database with client credentials configured (mTLS).
- Connect Oracle SQLcl Cloud Without a Wallet
  SQLcl is a command-line interface used to enter SQL commands. You can use SQLcl to connect to an Autonomous Database with TLS authentication without a wallet.

Connect Oracle SQL Developer with a Wallet (mTLS)

Oracle SQL Developer is a free integrated development environment that simplifies the development and management of Autonomous Database.

SQL Developer can connect to Autonomous Database and contains enhancements for key Autonomous Database features. You can download the latest version of Oracle SQL Developer for your platform from the **Download** link on this page: Oracle SQL Developer.

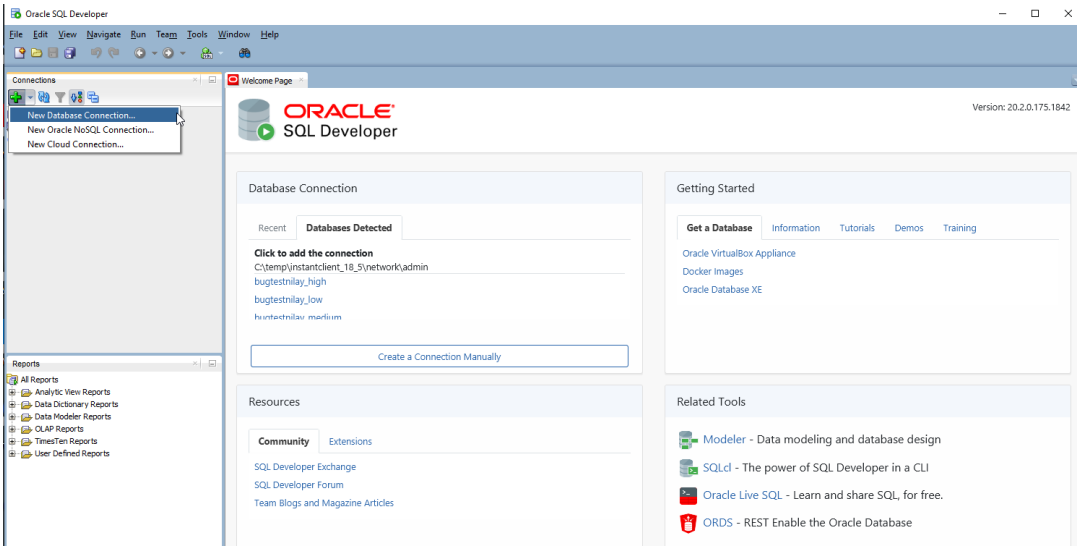
For connecting with mTLS authentication, Oracle SQL Developer provides support for wallet files using the **Cloud Wallet** Connection Type. Oracle recommends that you use version 18.2 (or later); however, earlier versions of SQL Developer will work with Autonomous Database using an Oracle Wallet.

For connecting with TLS authentication, Oracle SQL Developer provides support using the **Custom JDBC** Connection Type. See Connect with Oracle SQL Developer with TLS Authentication for details on connecting using TLS authentication.

To create a new mTLS connection to Autonomous Database, do the following:

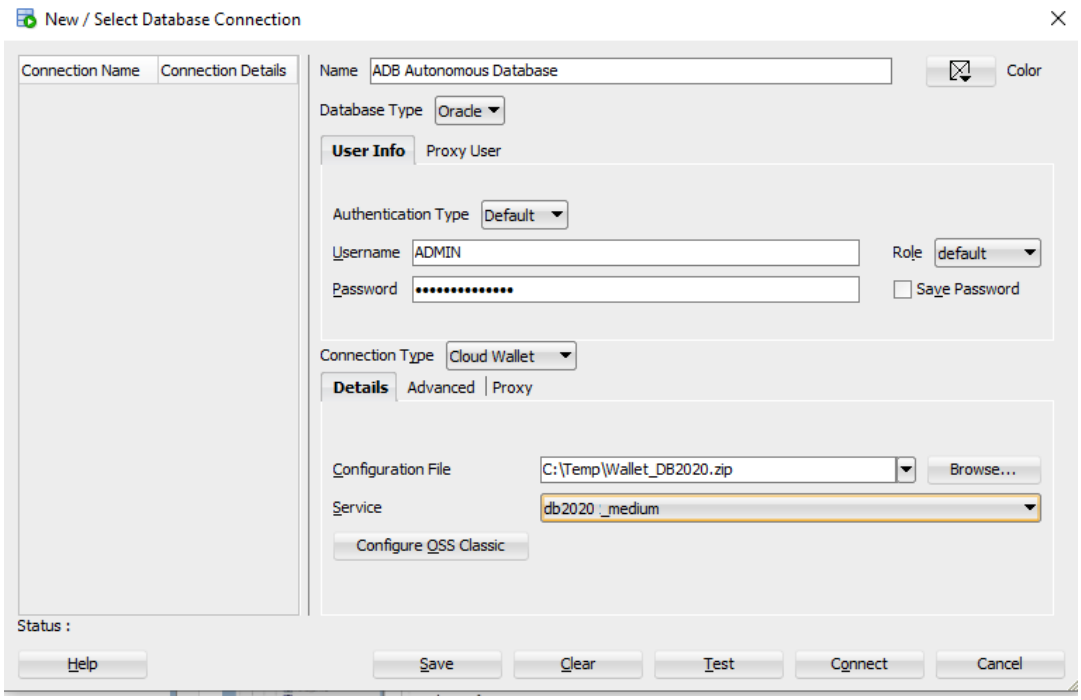
Obtain your credentials to access Autonomous Database. For more information, see Download Client Credentials (Wallets).

1. Start Oracle SQL Developer and in the connections panel, right-click **Connections** and select **New Database Connection...**.



2. Choose the Connection Type **Cloud Wallet**.
3. Enter the following information:
   - **Connection Name**: Enter the name for this connection.
   - **Username**: Enter the database username. You can either use the default administrator database account (`ADMIN`) provided as part of the service or create a new schema, and use it.
   - **Password**: Enter the password for the database user.
   - **Connection Type**: Select **Cloud Wallet** (if you are using SQL Developer 18.2, this is **Cloud PDB**)
   - **Configuration File** : Click **Browse**, and select the client credentials zip file.
   - **Service**: Enter the database TNS name. The client credentials file includes a `tnsnames.ora` file that provides database TNS names with corresponding services.

> **Note:**
>
> Versions of SQL Developer before 18.2 require that you enter a **Keystore Password**.



4. Click **Connect** to connect to the database.

> **Note:**
>
> If you are using Microsoft Active Directory, then for **Username** enter the Active Directory "*AD_domain\AD_username*" (you may include double quotes), and for the **Password**, enter the password for the Active Directory user. See Use Microsoft Active Directory with Autonomous Database for more information.

Connect Oracle SQL Developer Without a Wallet

Oracle SQL Developer is a free integrated development environment that simplifies the development and management of Autonomous Database. Oracle SQL Developer provides support for connecting using TLS authentication without a wallet.

> **Note:**
>
> See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for information on allowing TLS connections.

To create a new TLS connection to Autonomous Database:

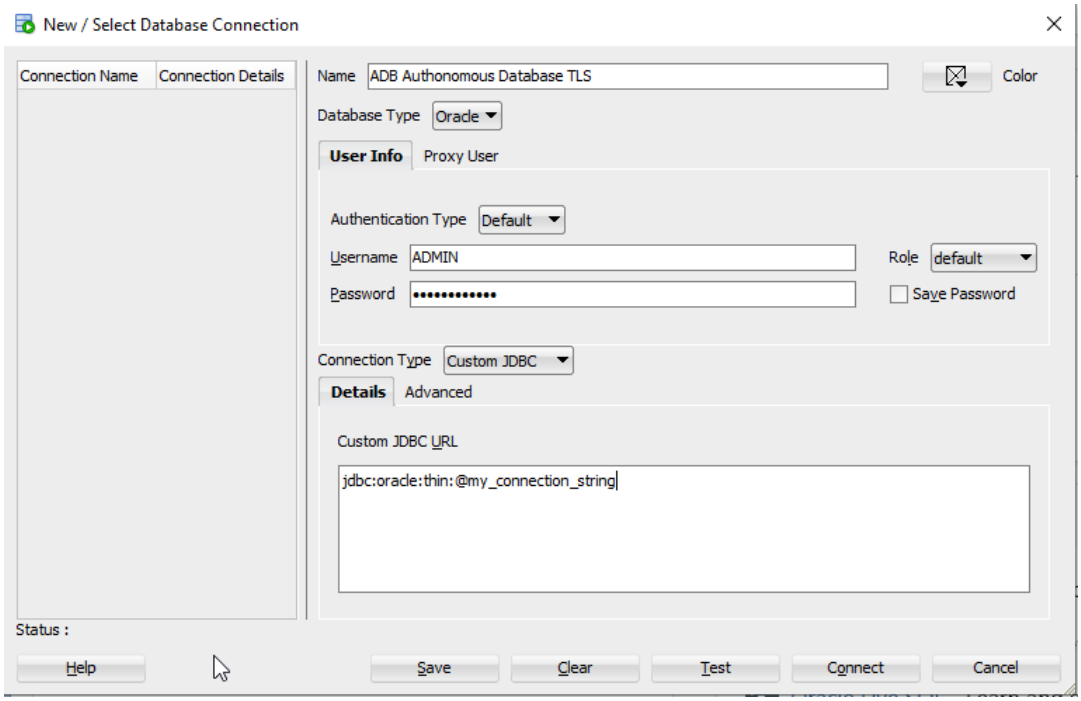
1. Copy a connection string for the Autonomous Database.

   To connect with TLS authentication copy a TLS connection string. On the Database Connection page, under **TLS Authentication**, select **TLS** to view the connection strings for connecting with TLS authentication.

See View TNS Names and Connection Strings for an Autonomous Database Instance for information on viewing and copying connection stings.

2. Start Oracle SQL Developer and in the connections panel, right-click **Connections** and select **New Database Connection...**.
3. Choose the Connection Type **Custom JDBC**.
4. Enter the following information:
   - **Name**: Enter the name for this connection.
   - **Username**: Enter the database username. You can either use the default administrator database account `ADMIN` provided as part of the service or create a new schema, and use it.
   - **Password**: Enter the password for the database user.
   - **Connection Type**: Select **Custom JDBC**.
   - **Custom JDBC URL**: Enter the following:

     `jdbc:oracle:thin:@` followed by the connection string you copied in step one.



For example, a sample value for the Custom JDBC URL field is:

```
jdbc:oracle:thin:@(description= (retry_count=20)(retry_delay=3)
(address=(protocol=tcps)
(port=1521)(host=adb-preprod.us-phoenix-1.oraclecloud.com))
(connect_data=(service_name=u9adutfb2_fmexample1_medium.adb.oraclecloud.com
))
(security=(ssl_server_dn_match=yes)))
```

When you copy the connection string, the values for *`region`* and *`databasename`* are for your Autonomous Database instance.

5. Click **Connect** to connect to the database.

> **Note:**
>
> If you are using Microsoft Active Directory, then for **Username** enter the Active Directory "*AD_domain\AD_username*" (you may include double quotes), and for the **Password**, enter the password for the Active Directory user. See Use Microsoft Active Directory with Autonomous Database for more information.

Connect SQL*Plus with a Wallet (mTLS)

SQL*Plus is a command-line interface used to enter SQL commands. SQL*Plus connects to an Oracle database.

To install and configure the client and connect to the Autonomous Database using SQL*Plus with client credentials (mTLS), do the following:

1. Prepare for Oracle Call Interface (OCI), ODBC and JDBC OCI Connections. See Prepare for Oracle Call Interface (OCI), ODBC, and JDBC OCI Connections with Wallets (mTLS).
2. Connect using a database user, *password*, and database TNS name provided in the `tnsnames.ora` file.

   For example:

```
sqlplus adb_user@db2022adb_medium

SQL*Plus: Release 19.0.0.0.0 - Production on Mon Nov 23 15:08:48 2020
Version 19.8.0.0.0

Copyright (c) 1982, 2020, Oracle.  All rights reserved.

Enter password:
Last Successful login time: Wed Nov 18 2020 12:36:56 -08:00

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.5.0.0.0

SQL>
```

> **Notes:**
>
> - The Oracle Wallet is transparent to SQL*Plus because the wallet location is specified in the `sqlnet.ora` file. This is true for any Oracle Call Interface (OCI), ODBC, or JDBC OCI connection.
> - If you are connecting to a database using Microsoft Active Directory credentials, then connect using an Active Directory user name in the form of "*AD_domain\AD_username*" (double quotes must be included), and Active Directory user password. See Use Microsoft Active Directory with Autonomous Database for more information.

Connect SQL*Plus Without a Wallet

SQL*Plus is a command-line interface used to enter SQL commands. SQL*Plus connects to an Oracle database.

> **Note:**
>
> See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for information on allowing TLS connections.

To install and configure the client and connect to the Autonomous Database using SQL*Plus with TLS authentication, do the following:

1. Prepare for Oracle Call Interface (OCI) connections.

   You can use TLS authentication without a wallet in SQL*Plus if you are using the following client versions:

   - Oracle Instant Client/Oracle Database Client 19.13 - only on Linux x64
   - Oracle Instant Client/Oracle Database Client 19.14 (or later) and 21.5 (or later) - only on Linux x64 and Windows

   See Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections Using TLS Authentication for more information.

2. Copy a connection string for the Autonomous Database.

   To connect with TLS authentication copy a TLS connection string. On the Database Connection page, under **TLS Authentication**, select **TLS** to view the connection strings for connecting with TLS authentication.

   See View TNS Names and Connection Strings for an Autonomous Database Instance for information on viewing and copying connection stings.

   See Predefined Database Service Names for Autonomous Database for information on the different databases services for each connection string.

3. Connect using a database user and *password*, and provide the connection string you copied in Step 2.

   On UNIX/Linux start `sqlplus` with the connection string, enclosed in quotes on the command line, as follows:

   ```
   sqlplus username/password@'my_connect_string'
   ```

   For example (for clarity line breaks added):

   ```
   sqlplus adb_user/password@'(description= (retry_count=20)(retry_delay=3)
   (address=(protocol=tcps)
   (port=1521)(host=adb.region.oraclecloud.com))
   (connect_data=(service_name=u9adutfb2ba8x4d_database_medium.adb.oraclecloud
   .com))
   (security=(ssl_server_dn_match=yes))'
   ```

On Windows, start `sqlplus` with the database user and password with the copied connection string, as follows (as compared to UNIX/Linux, on Windows do not surround the connection string with quotes):

```
sqlplus username/password@my_connect_string
```

For example (for clarity line breaks added in the connection string):

```
sqlplus adb_user/password@(description= (retry_count=20)(retry_delay=3)
(address=(protocol=tcps)
(port=1521)(host=adb.region.oraclecloud.com))
(connect_data=(service_name=u9adutfb2ba8x4d_database_medium.adb.oraclecloud
.com))
(security=(ssl_server_dn_match=yes))
```

> **Note:**
>
> If you are connecting to an Autonomous Database instance using Microsoft Active Directory credentials, then connect using an Active Directory user name in the form of "*AD_domain\AD_username*" (double quotes must be included), and Active Directory user password. See Use Microsoft Active Directory with Autonomous Database for more information.

Connect Oracle SQLcl Cloud with a Wallet (mTLS)

SQLcl is a command-line interface used to enter SQL commands. You can use SQLcl to connect to an Autonomous Database with client credentials configured (mTLS).

You can use SQLcl version 4.2 or later with Autonomous Database. Download SQLcl from oracle.com.

SQLcl can connect to an Autonomous Database instance using either an Oracle Call Interface (OCI) or a JDBC thin connection.

- If you use Oracle Call Interface (OCI), prepare for OCI, ODBC and JDBC OCI Connections. See Prepare for Oracle Call Interface (OCI), ODBC, and JDBC OCI Connections.
- If you use JDBC Thin, prepare for JDBC Thin Connections. See Prepare for JDBC Thin Connections.

SQLcl with Oracle Call Interface

To connect using Oracle Call Interface, use the `-oci` option, supply the database user name, a password, and the database service name provided in the `tnsnames.ora` file. For example:

```
sql -oci

SQLcl: Release 22.1 Production on Fri May 06 16:07:46 2022

Copyright (c) 1982, 2022, Oracle.  All rights reserved.

Username? (''?) adb_user@db2022adb_medium
Password? (**********?) ***************
```

```
Connected.
SQL>
```

When connecting using Oracle Call Interface, the Oracle Wallet is transparent to SQLcl.

SQLcl with a JDBC Thin Connection

To connect using a JDBC Thin connection, first configure the SQLcl cloud configuration and then connect to the database.

1. Start SQLcl with the /nolog option.

   ```
   sql /nolog
   ```

2. Configure the SQLcl session to use your Oracle Wallet:

   ```
   SQL> set cloudconfig directory/client_credentials.zip
   ```

3. Connect to the database:

   ```
   SQL> connect username@servicename
   password
   ```

   To avoid the prompt, connect and supply the password inline:

   ```
   SQL> connect username/password@servicename
   ```

   For example:

   ```
   sql /nolog

   SQLcl: Release 22.1 Production on Fri May 06 14:48:26 2022

   Copyright (c) 1982, 2022, Oracle.  All rights reserved.

   SQL> set cloudconfig /home/adb/Wallet_db2022ADB.zip

   SQL> connect adb_user@db2022adb_medium

   Password? (**********?) ***************
   Connected.
   SQL>
   ```

SQLcl with a JDBC Thin Connection with an HTTP Proxy

1. Start SQLcl with the /nolog option.

   ```
   sql /nolog
   ```

2. Configure the SQLcl session to use a proxy host and your Oracle Wallet:

   ```
   SQL> set cloudconfig -proxy=proxyhost:port directory/client_credentials.zip
   ```

3. Connect to the database.

   ```
   SQL> connect username@servicename
   password
   ```

   To avoid the prompt, connect and supply the password inline:

   ```
   SQL> connect username/password@servicename
   ```

For example:

```
sql /nolog

SQLcl: Release 22.1 Production on Fri May 06 11:59:38 2022

Copyright (c) 1982, 2022, Oracle.  All rights reserved.
SQL> set cloudconfig -proxy=http://myproxyhost.com:80 /home/adb/
Wallet_db2022.zip

SQL> connect adb_user@db2022adb_medium

Password? (**********?) ***************
Connected.
SQL>
```

> **Note:**
>
> If you are connecting to Autonomous Database using Microsoft Active Directory credentials, then connect using an Active Directory user name in the form of "*AD_domain\AD_username*" (double quotes must be included), and Active Directory user password. See Use Microsoft Active Directory with Autonomous Database for more information.

For more information, on the connection types specified in `tnsnames.ora`, see Manage Concurrency and Priorities on Autonomous Database.

For information on SQLcl, see Oracle SQLcl.

Connect Oracle SQLcl Cloud Without a Wallet

SQLcl is a command-line interface used to enter SQL commands. You can use SQLcl to connect to an Autonomous Database with TLS authentication without a wallet.

> **Note:**
>
> See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for information on allowing TLS connections.

You can use SQLcl version 4.2 or later with Autonomous Database. Download SQLcl from oracle.com.

If you use JDBC Thin Driver, prepare for JDBC Thin connections. See Prepare for JDBC Thin Connections.

To connect using a JDBC Thin Driver with TLS authentication, do the following to connect to the database.

1. Copy a connection string for the Autonomous Database.

To connect with TLS authentication copy a TLS connection string. On the Database Connection page, under **TLS Authentication**, select **TLS** to view the connection strings for connecting with TLS authentication.

See View TNS Names and Connection Strings for an Autonomous Database Instance for information on viewing and copying connection stings.

See Predefined Database Service Names for Autonomous Database for information on the different databases services for each connection string.

2. Start SQLcl and connect to the database:

   On UNIX/Linux start `sql` with the connection string, enclosed in quotes on the command line, as follows:

   ```
   sql username/password@'my_connect_string'
   ```

   For example (for clarity line breaks added):

   ```
   $ sql admin/password@'(description= (retry_count=20)(retry_delay=3)
   (address=(protocol=tcps)(port=1521)(host=adb.region.oraclecloud.com))
   (connect_data=(service_name=u9adutfb2ba8x4d_database_medium.adb.oraclecloud
   .com))
   (security=(ssl_server_dn_match=yes)))'

   SQLcl: Release 21.2 Production on Thu Sep 16 10:43:00 2021
   Copyright (c) 1982, 2021, Oracle. All rights reserved.

   Last Successful login time: Thu Sep 16 2021 10:43:01 -07:00

   Connected to:
   Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
   Version 19.12.0.1.0

   SQL>
   ```

   On Windows, start `sql` with the `/nolog` option and then connect with the copied connection string, as follows (as compared to UNIX/Linux, on Windows do not surround the connection string with quotes):

   ```
   > sql /nolog

   SQLcl: Release 21.2 Production on Fri Sep 17 10:15:01 2021
   Copyright (c) 1982, 2021, Oracle.  All rights reserved.
   SQL> conn username/password@my_connect_string
   ```

   For example (for clarity line breaks are added):

   ```
   > sql admin/password@(description= (retry_count=20)(retry_delay=3)
   (address=(protocol=tcps)(port=1521)(host=adb.region.oraclecloud.com))
   (connect_data=(service_name=u9adutfb2ba8x4d_database_medium.adb.oraclecloud
   .com))
   (security=(ssl_server_dn_match=yes)))

   SQLcl: Release 21.2 Production on Thu Sep 16 10:43:00 2021
   Copyright (c) 1982, 2021, Oracle. All rights reserved.
   ```

```
Last Successful login time: Thu Sep 16 2021 10:43:01 -07:00

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.12.0.1.0

SQL>
```

> **Note:**
>
> If you are connecting to Autonomous Database using Microsoft Active Directory credentials, then connect using an Active Directory user name in the form of "*AD_domain\AD_username*" (double quotes must be included), and Active Directory user password. See Use Microsoft Active Directory with Autonomous Database for more information.

For information on SQLcl, see Oracle SQLcl.

Connect with Built-In Oracle Database Actions

You can access Database Actions from Autonomous Database. Database Actions provides development tools, data tools, administration, and monitoring features for Autonomous Database. Using Database Actions you can run SQL statements, queries, and scripts in a worksheet.

- About Database Actions (SQL Developer Web)
  Database Actions provides a web-based interface with development, data tools, administration, monitoring, and download features for Autonomous Database.
- Access Database Actions as ADMIN
  Database Actions (also known as SQL Developer Web) is bundled with each Autonomous Database instance.
- Provide Database Actions Access to Database Users
  The ADMIN user provides access to Database Actions for other database users.
- Required Roles to Access Tools from Database Actions
  Lists the database roles required to use the built-in Autonomous Database tools.
- Access Database Actions, Oracle APEX, Oracle REST Data Services, and Developer Tools Using a Vanity URL
  By default you access Oracle APEX apps, REST endpoints, and developer tools on Autonomous Database using the `oraclecloudapps.com` domain name. You can optionally configure a vanity URL or custom domain name that is easy to remember to help promote your brand identity.

About Database Actions (SQL Developer Web)

Database Actions provides a web-based interface with development, data tools, administration, monitoring, and download features for Autonomous Database.

The following table lists the main features of Database Actions. Database Actions on your Autonomous Database instance may include additional cards if you download applications from Oracle Cloud Marketplace.

Feature Area	Database Actions Cards
Development	SQL, Data Modeler, REST, JSON, Charts, Scheduling, Oracle Machine Learning, Graph Studio, and Oracle APEX
Data Studio	Data Load, Catalog, Data Insights, Data Transforms, and Data Analysis
Administration	Database Users, APEX Workspaces, Data Pump, Download Client Credentials, and Set Resource Management Rules
Monitoring	Performance Hub and Database Dashboard
Downloads	Download Oracle Instant Client and Download SODA Drivers
Related Services	Restful Data Services (ORDS) and SODA and Access Oracle Machine Learning Restful Services

See About Database Actions in *Using Oracle Database Actions* for more information.

Access Database Actions as ADMIN

Database Actions (also known as SQL Developer Web) is bundled with each Autonomous Database instance.

Database Actions runs in Oracle REST Data Services and access is provided through schema-based authentication. To use Database Actions, you must sign in as a database user whose schema is enabled for Database Actions. By default the ADMIN user is enabled to access Database Actions.

See Provide Database Actions Access to Database Users to enable another database user's schema to access Database Actions.

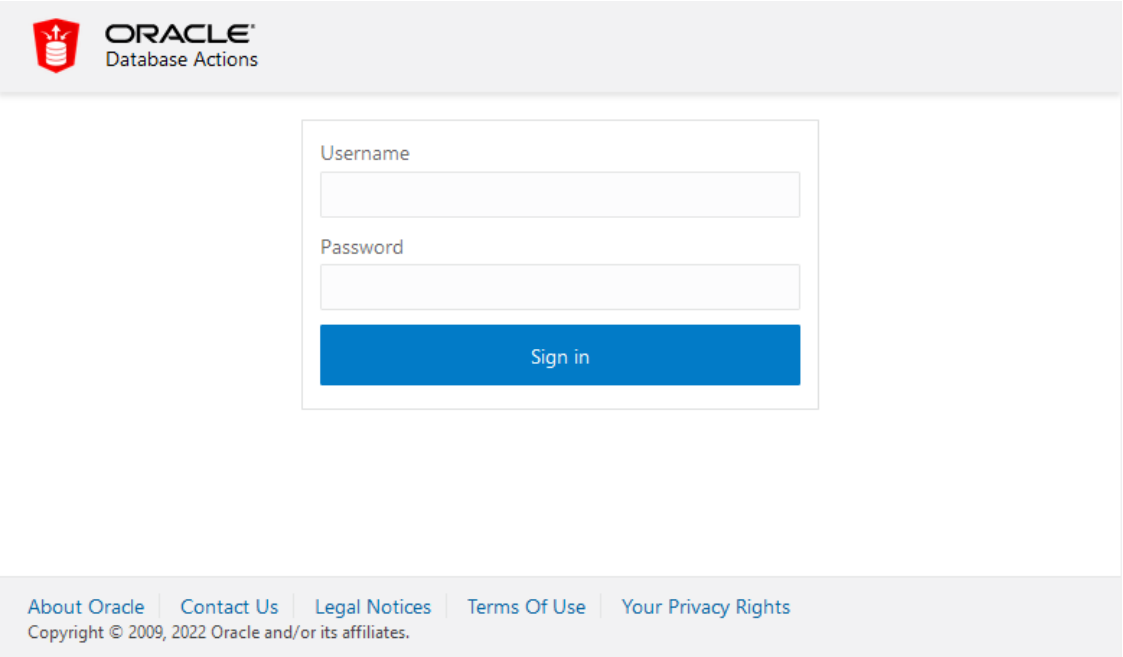
> **Note:**
>
> If your Autonomous Database is configured to use a Private Endpoint, then you can only access Database Actions from clients in the same Virtual Cloud Network (VCN). See Configure Network Access with Private Endpoints for more information.

To access Database Actions from the Oracle Cloud Infrastructure Console:

1. On the Autonomous Database Details page click the **Database actions** dropdown list.
2. Select a quick link to go directly to a quick link action or select **View all database actions** to access the full Database Actions Launchpad.

   For example, click **SQL** to use a SQL Worksheet. On the SQL Worksheet you can use the **Consumer Group** drop-down list to select the consumer group to run your SQL or PL/SQL code. See Executing SQL Statements in the Worksheet Editor for more information.

Depending on your browser, if the Console cannot access the database as ADMIN you will be prompted for your database ADMIN username and password.



Provide Database Actions Access to Database Users

The ADMIN user provides access to Database Actions for other database users.

Database users who are not service administrators do not have access to the Oracle Cloud Infrastructure Console. The ADMIN user provides access to Database Actions as follows:

- Use Database Actions to create a user and assign roles to the user. If the user already exists, check that Web Access is selected for the schema (with Web Access selected, the user's card shows **REST Enabled**).

  See Create Users on Autonomous Database for information on adding database users.

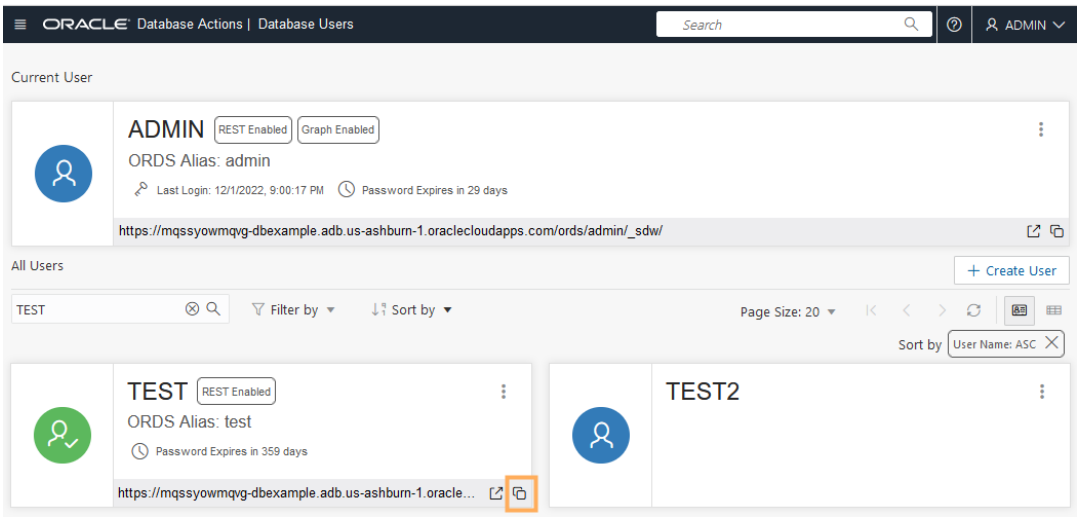
  See Required Roles to Access Tools from Database Actions for information on required roles for Database Actions.
- Provide the user with a URL to access Database Actions.

After adding a user and selecting Web Access, the ADMIN provides a user with the URL to access Database Actions:

1. In Database Actions, click ☰ to show the available actions.
2. Under **Administration** select **Database Users**.

   This displays information about users, such as user names, whether a user is REST Enabled, and the last login date and time. On a user's card, the icon on the left displays the user status with one of the following colors: green (Open), blue (Locked), or red (Expired).

   The default view is Card View. You can select either grid view or card view by clicking the **Card View** or **Grid View** icons.
3. A URL is displayed in the user's card only if the user is REST Enabled. It provides the URL to access Database Actions. Click ⧉ to copy the URL to the clipboard.



4. Provide the user with the URL you copied.

After you provide the URL to a user, to access Database Actions the user pastes the URL into their browser and then enters their Username and Password in the Sign-in dialog.

See Manage Users and User Roles on Autonomous Database - Connecting with Database Actions for more information.

Required Roles to Access Tools from Database Actions

Lists the database roles required to use the built-in Autonomous Database tools.

When you connect to Database Actions as the ADMIN user, many database roles are set to allow you to access the available Autonomous Database tools. When you connect to Database Actions as a non-ADMIN user, some of the Database Actions cards for Autonomous Database tools are hidden if you do not have required permissions.

As the ADMIN user, set the appropriate roles to allow user access to Autonomous Database tools. See Manage User Roles and Privileges on Autonomous Database for more information.

Autonomous Database Tool	Required Role	More Information
Oracle Machine Learning (OML)	`OML_DEVELOPER`	Oracle Machine Learning
Graph Studio	`GRAPH_DEVELOPER`	Using Oracle Graph with Autonomous Database
Data Studio (including: Catalog, Data Load, Data Analysis, Data Insights)	`DWROLE`	The Data Studio Overview Page
Data Transforms	`DATA_TRANSFORM_USER`	

Access Database Actions, Oracle APEX, Oracle REST Data Services, and Developer Tools Using a Vanity URL

By default you access Oracle APEX apps, REST endpoints, and developer tools on Autonomous Database using the `oraclecloudapps.com` domain name. You can optionally configure a vanity URL or custom domain name that is easy to remember to help promote your brand identity.

After you acquire a desired domain name and matching SSL certificate from a vendor of your choice, deploy an Oracle Cloud Infrastructure Load Balancer in your Virtual Cloud Network (VCN) using your Autonomous Database as the backend. Your Autonomous Database instance must be configured with a private endpoint in the same VCN. See Configuring Network Access with Private Endpoints for more information.

To learn more, see the following:

- Introducing Vanity URLs for APEX and ORDS on Oracle Autonomous Database
- Automate Vanity URL Configuration Using Terraform

Connect with JDBC Thin Driver

Autonomous Database mandates a secure connection that uses Transport Layer Security (TLSv1.2).

Java applications that use JDBC Thin driver connect with one of the following:

- **Mutual TLS (mTLS) Authentication**: requires either Oracle Wallet or Java KeyStore (JKS) where both the client and Autonomous Database authenticate each other.

  The wallet and keystore files are included in the client credentials `.zip` file that is available by clicking **Database connection** on the Oracle Cloud Infrastructure Console
- **TLS Authentication**: The client computer matches the server's CA root certificate against the client's list of trusted CAs. If the issuing CA is trusted, the client verifies that the certificate is authentic. This allows the client and Autonomous Database to establish the encrypted connection before exchanging any messages.

Topics

JDBC Thin Connections with a Wallet (mTLS)

Autonomous Database mandates a secure connection that uses Transport Layer Security (TLSv1.2). Depending on the network configuration options, Autonomous Database supports mTLS and TLS authentication.

> **Note:**
>
> If you use TLS, instead of mTLS, for your connections using JDBC Thin Driver with JDK8u162 or higher, a wallet is not required.
>
> TLS connections are enabled for the following network configurations:
>
> - **Private endpoint access only**: network configuration with a private endpoint
> - **Secure access from allowed IPs and VCNs only**: configuration with an Access Control List (ACL)
>
>   If your Autonomous Database is on a public endpoint without any ACL, you can add `0.0.0.0/0` as your CIDR ACL and enable TLS authentication. Adding `0.0.0.0/0` as your CIDR ACL is identical to having your Autonomous Database on public endpoint with no ACL.
>
> See About TLS Authentication for more information.

- JDBC Thin Driver Connection Prerequisites Connections with Wallets (mTLS)
  Applications that use JDBC Thin driver support TLS and mutual TLS (mTLS) authentication. Using mTLS authentication requires that you supply Oracle database credentials including the Oracle wallets or Java KeyStore (JKS) files when connecting to the database.
- Using a JDBC URL Connection String with JDBC Thin Driver and Wallets
  The connection string is found in the file `tnsnames.ora` which is part of the client credentials download. The `tnsnames.ora` file contains the predefined service names. Each service has its own TNS alias and connection string.
- Using a JDBC Connection with 18.3 JDBC Driver
  Applications that use JDBC Thin driver can connect to Autonomous Databases using either Oracle Wallets or Java KeyStore (JKS).
- Connecting Using JDBC Thin Driver 12.2 or Older
  If you are using the JDBC driver 12.2.0.2 or older, set the Java properties prior to starting the application. Usually you set the properties in the application's startup script.
- JDBC Thin Connections with an HTTP Proxy
  If the client is behind a firewall and your network configuration requires an HTTP proxy to connect to the internet, you need to use the JDBC Thin Client 18.1 or higher which enables connections through HTTP proxies.

JDBC Thin Driver Connection Prerequisites Connections with Wallets (mTLS)

Applications that use JDBC Thin driver support TLS and mutual TLS (mTLS) authentication. Using mTLS authentication requires that you supply Oracle database credentials including the Oracle wallets or Java KeyStore (JKS) files when connecting to the database.

Perform the following steps before connecting to the database:

1. **Provision Autonomous Database**: Create a database and obtain your database credentials (username and password).

   See Provision Autonomous Database for more information.

2. For mutual TLS connections, **Download Client Credentials**: Unzip the `wallet_`*databasename*`.zip` to a secure location. Make sure that only authorized users have access to these files.

   See Download Client Credentials (Wallets) for information on downloading client credentials for Autonomous Database.

3. **Verify your JDK version for security**: If you are using JDK11, JDK10, or JDK9 then you don't need to do anything for this step. If your JDK version is less than JDK8u162 then you need to download the JCE Unlimited Strength Jurisdiction Policy Files. Refer to the `README` file for installation notes. Download the JCE files from Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8 Download.

4. **Check JDBC Driver Version**: Download a supported JDBC Thin driver (`ojdbc8.jar` and `ucp.jar`). You also need the additional jars: `oraclepki.jar`, `osdt_core.jar`, and `osdt_cert.jar` for use with Oracle wallets.

   Supported versions are:

   - **JDBC Thin**: 11.2.0.4 (or later with one-off patch for Bug 28492769), 12.2 (or later with one-off patch for Bug 28492769), 18 (base release or later with one-off patch for Bug 28492769), 19 (base release or later), or 21 (base release or later)

   For applications that use the Universal Connection Pool (UCP) feature of JDBC, it is highly recommended to use 19.13 or higher or 21.3 or higher versions of the JDBC driver. These versions include proper draining behavior to minimize impact to applications when planned maintenance is performed in Autonomous Database. UCP will replenish connections in the pool proactively so that active connections are not impacted by maintenance.

   For older versions of the driver, a patch for bug 31112088 can also be requested by filing a Service Request.

   Download supported versions: Oracle Database JDBC driver and Companion Jars Downloads .

Using a JDBC URL Connection String with JDBC Thin Driver and Wallets

The connection string is found in the file `tnsnames.ora` which is part of the client credentials download. The `tnsnames.ora` file contains the predefined service names. Each service has its own TNS alias and connection string.

A sample entry, with `dbname_high` as the TNS alias and a connection string in `tnsnames.ora` follows:

```
dbname_high= (description=
      (address=(protocol=tcps)(port=1522)(host=adb.example.oraclecloud.com))
```

```
      (connect_data=(service_name=dbname_high.oraclecloud.com))
(security=(ssl_server_dn_match=yes)))
```

Set the location of `tnsnames.ora` with the property `TNS_ADMIN` in one of the following ways:

- As part of the connection string (only with the 18.3 or newer JDBC driver)
- As a system property, `-Doracle.net.tns_admin`
- As a connection property (`OracleConnection.CONNECTION_PROPERTY_TNS_ADMIN`)

Using the 18.3 JDBC driver, the connection string includes the TNS alias and the `TNS_ADMIN` connection property.

Sample connection string using 18.3 JDBC driver (Linux):

```
DB_URL="jdbc:oracle:thin:@dbname_high?TNS_ADMIN=/Users/test/wallet_dbname"
```

Sample connection string using 18.3 JDBC driver (Windows):

```
DB_URL="jdbc:oracle:thin:@dbname_high?TNS_ADMIN=C:\\Users\\test\
\wallet_dbname"
```

The `TNS_ADMIN` connection property specifies the following:

- The location of `tnsnames.ora`.
- The location of Oracle Wallet (`ewallet.sso`, `ewallet.p12`) or Java KeyStore (JKS) files (`truststore.jks`, `keystore.jks`).
- The location of `ojdbc.properties`. This file contains the connection properties required to use Oracle Wallets or Java KeyStore (JKS).

> **Note:**
>
> If you are using 12.2.0.1 or older JDBC drivers, then the connection string contains only the TNS alias. To connect using older JDBC drivers:
>
> - Set the location of the `tnsnames.ora`, either as a system property with `-Doracle.net.tns_admin` or as a connection property (`OracleConnection.CONNECTION_PROPERTY_TNS_ADMIN`).
> - Set the wallet or JKS related connection properties in addition to `TNS_ADMIN`.
>
> For example, in this case you set the TNS alias in the `DB_URL` without the `TNS_ADMIN` part as:
>
> ```
> DB_URL=”jdbc:oracle:thin:@dbname_high”
> ```

See Predefined Database Service Names for Autonomous Database for more details.

Using a JDBC Connection with 18.3 JDBC Driver

Applications that use JDBC Thin driver can connect to Autonomous Databases using either Oracle Wallets or Java KeyStore (JKS).

Using Oracle Wallet

To use Java and the 18.3 JDBC Thin Driver to connect to Autonomous Database with the Oracle Wallet, do the following:

1. **Make sure that the prerequisites are met:** See JDBC Thin Driver Connection Prerequisites Connections with Wallets (mTLS) for more information.
2. **Verify the connection**: You can either use a Java program, a servlet, or IDEs to verify the connection to the database. A simple test is to download DataSourceSample.java or UCPSample.java from JDBC code samples and update the connection URL to have the required TNS alias and pass `TNS_ADMIN`, providing the path for `tnsnames.ora` and the wallet files. Also, in the sample source code update the database username and password. For example:

   ```
   DB_URL="jdbc:oracle:thin:@dbname_high?TNS_ADMIN=/Users/test/wallet_dbname"
   ```

   > **Note:**
   >
   > If you are using Microsoft Active Directory with a database, then in the sample source code update the username with the Active Directory username and update the password with the Active Directory user password. See Use Microsoft Active Directory with Autonomous Database for more information.

3. **Set the wallet location**: The properties file `ojdbc.properties` is pre-loaded with the wallet related connection property.

   ```
   oracle.net.wallet_location=(SOURCE=(METHOD=FILE)(METHOD_DATA=(DIRECTORY=${TNS_ADMIN})))
   ```

   > **Note:**
   >
   > You do not modify the file `ojdbc.properties`. The value of `TNS_ADMIN` determines the wallet location.

4. **Compile and Run:** Compile and run the sample to get a successful connection. Make sure you have `oraclepki.jar` , `osdt_core.jar`, and `osdt_cert.jar`, in the `classpath`. For example:

   ```
   java -classpath
         ./lib/ojdbc8.jar:./lib/ucp.jar:./lib/oraclepki.jar:./lib/
   osdt_core.jar:./lib/osdt_cert.jar:. UCPSample
   ```

> **Note:**
>
> The auto-login wallet part of Autonomous Database downloaded client credentials zip file removes the need for your application to use username/password authentication.

Using Java KeyStore

To use Java and the 18.3 JDBC Thin Driver to connect to Autonomous Database with Java KeyStore (JKS), do the following:

1. **Make sure that the prerequisites are met:** See JDBC Thin Driver Connection Prerequisites Connections with Wallets (mTLS) for more information.
2. **Ready the database details**: You can either use a Java program, a servlet, or IDEs to check the connection to your database. A simple test is to download DataSourceSample.java or UCPSample.java from JDBC code samples. In this sample, use the connection URL as shown. Note that the connection `DB_URL` contains the TNS alias, for example, `dbname_high` present in `tnsnames.ora`. You can provide the path for `tnsnames.ora` file through `TNS_ADMIN` property as shown in the URL. Make sure to use the database username and password related to your database.

   ```
   DB_URL="jdbc:oracle:thin:@dbname_high?TNS_ADMIN=/Users/test/wallet_dbname"
   ```

   > **Note:**
   >
   > If you are using Microsoft Active Directory with Autonomous Database, then make sure to change the sample source code to use the Active Directory username and the Active Directory user password. See Use Microsoft Active Directory with Autonomous Database for more information.

3. **Set JKS related connection properties**: Add the JKS related connection properties to `ojdbc.properties` file. The keyStore and truststore password are the password specified when you downloading the client credentials `.zip` file.

   To use SSL connectivity instead of Oracle Wallet, specify the keystore and truststore files and their respective password in the `ojdbc.properties` file as follows:

   ```
   # Properties for using Java KeyStore (JKS)
   oracle.net.ssl_server_dn_match=true
   javax.net.ssl.trustStore==${TNS_ADMIN}/truststore.jks
   javax.net.ssl.trustStorePassword=password
   javax.net.ssl.keyStore==${TNS_ADMIN}/keystore.jks
   javax.net.ssl.keyStorePassword=password
   ```

> **Note:**
>
> Make sure to comment the wallet related property in `ojdbc.properties`. For example:
>
> ```
> # Property for using Oracle Wallets
> # oracle.net.wallet_location=(SOURCE=(METHOD=FILE)
> (METHOD_DATA=(DIRECTORY=${TNS_ADMIN})))
> ```

4. **Compile and Run:** Compile and run the sample to get a successful connection. For example:

   ```
   java -classpath ./lib/ojdbc8.jar:./lib/ucp.jar UCPSample
   ```

Connecting Using JDBC Thin Driver 12.2 or Older

If you are using the JDBC driver 12.2.0.2 or older, set the Java properties prior to starting the application. Usually you set the properties in the application's startup script.

If you are not able to use the latest 18.3 JDBC drivers, then you can connect to Autonomous Database using 12.2.0.2 or other older JDBC drivers. The 12.2 or older JDBC drivers do not support the `ojdbc.properties` file. With older JDBC driver versions, you need to pass wallets or JKS related properties either as system properties or as connection properties to establish a connection.

Using Oracle Wallet

To use Java and the 12.2 or older JDBC Drivers to connect to Autonomous Database with the Oracle Wallet, do the following:

1. **Make sure that the prerequisites are met:** See JDBC Thin Driver Connection Prerequisites Connections with Wallets (mTLS) for more information.
2. **Verify the connection**: You can either use a Java program, a servlet, or IDEs to verify the connection to the database. A simple test is to download DataSourceSample.java or UCPSample.java from JDBC code samples and update the connection URL to have the required TNS alias. Also, update the sample source code to use the database username and password. For example:

   ```
   DB_URL="jdbc:oracle:thin:@dbname_high”
   ```

> **Note:**
>
> If you are using Microsoft Active Directory with Autonomous Database, then update the sample source code to use the Active Directory username and Active Directory user password. See Use Microsoft Active Directory with Autonomous Database for more information.

3. **Set the wallet location**: Add the `OraclePKIProvider` at the end of the provider list in the file `java.security` (this file is part of your JRE install located at `$JRE_HOME/jre/lib/security/java.security`) which typically looks like:

   ```
   security.provider.14=oracle.security.pki.OraclePKIProvider
   ```

4. **Compile and Run:** Compile and run the sample to get a successful connection. Make sure to have `oraclepki.jar` , `osdt_core.jar`, and `osdt_cert.jar`, in the `classpath`. Also, you need to pass the connection properties. Update the properties with the location where `tnsnames.ora` and wallet files are located.

   ```
   java -classpath
   ./lib/ojdbc8.jar:./lib/ucp.jar:./lib/oraclepki.jar:./lib/
   osdt_core.jar:./lib/osdt_cert.jar:.
   -Doracle.net.tns_admin=/users/test/wallet_dbname
   -Doracle.net.ssl_server_dn_match=true
   -Doracle.net.ssl_version=1.2   (Not required for 12.2)
   -Doracle.net.wallet_location= "(SOURCE=(METHOD=FILE)
   (METHOD_DATA=(DIRECTORY=/users/test/wallet_dbname)))"
   UCPSample
   ```

> **Note:**
>
> These are Windows system examples. Add a `\` continuation character if you are setting `-D` properties on multiple lines on UNIX (Linux or a Mac).

Using Java KeyStore

To use Java and the 12.2 or older JDBC Thin Drivers to connect to Autonomous Database with Java KeyStore (JKS), do the following:

1. **Make sure that the prerequisites are met:** See JDBC Thin Driver Connection Prerequisites Connections with Wallets (mTLS) for more information.
2. **Verify the connection**: You can either use a Java program, a servlet, or IDEs to verify the connection to the database. A simple test is to download DataSourceSample.java or UCPSample.java from JDBC code samples and update the connection URL to have the required TNS alias and pass `TNS_ADMIN`, providing the path for `tnsnames.ora` and update the connection URL to have the required TNS alias. Also, in the sample source code update the database username and password. For example:

   ```
   DB_URL="jdbc:oracle:thin:@dbname_high"
   ```

   > **Note:**
   >
   > If you are using Microsoft Active Directory with Autonomous Database, then update the sample source code to use the Active Directory username and Active Directory user password. See Use Microsoft Active Directory with Autonomous Database for more information.

3. **Compile and Run**: Compile and run the sample to get a successful connection. You need to pass the connection properties as shown. Update the properties with the location where

`tnsnames.ora` and JKS files are placed. If you want to pass these connection properties programmatically then refer to DataSourceForJKS.java. For example:

```
java
-Doracle.net.tns_admin=/users/test/wallet_dbname
-Djavax.net.ssl.trustStore=truststore.jks
-Djavax.net.ssl.trustStorePassword==========
-Djavax.net.ssl.keyStore=keystore.jks
-Djavax.net.ssl.keyStorePassword============
-Doracle.net.ssl_server_dn_match=true
-Doracle.net.ssl_version=1.2 // Not required for 12.2
```

JDBC Thin Connections with an HTTP Proxy

If the client is behind a firewall and your network configuration requires an HTTP proxy to connect to the internet, you need to use the JDBC Thin Client 18.1 or higher which enables connections through HTTP proxies.

To connect to Autonomous Database through an HTTPS proxy, open and update your `tnsnames.ora` file. Add the HTTP proxy hostname(https_proxy) and port (https_proxy_port) to the connection string. Replace the values with your HTTPS proxy information. For example:

1. Add the HTTP proxy hostname and port to the connection definitions in `tnsnames.ora`. You need to add the https_proxy and https_proxy_port parameters in the address section of connection definitions. For example, the following sets the HTTP proxy to `proxyhostname` and the HTTP proxy port to `80`; replace these values with your HTTP proxy information:

```
db2022adb_high =
       (description=
             (address=
                   (https_proxy=proxyhostname)(https_proxy_port=80)
(protocol=tcps)(port=1522)(host=adb.example.oraclecloud.com)
             )
             (connect_data=(service_name=db2022adb_high.adb.oraclecloud.com)
             )
             (security=security=(ssl_server_dn_match=yes)
             )
       )
```

> **Notes:**
>
> - JDBC Thin client versions earlier than 18.1 do not support connections through HTTP proxy.
> - Successful connection depends on specific proxy configurations and the performance of data transfers would depend on proxy capacity. Oracle does not recommend using this feature in Production environments where performance is critical.
> - Configuring `tnsnames.ora` for the HTTP proxy may not be enough depending on your organization's network configuration and security policies. For example, some networks require a username and password for the HTTP proxy.
> - In all cases, contact your network administrator to open outbound connections to hosts in the `oraclecloud.com` domain using the relevant port without going through an HTTP proxy.

JDBC Thin Connections Without a Wallet (TLS)

Autonomous Database mandates a secure connection that uses Transport Layer Security (TLSv1.2). Depending on the configuration options, Autonomous Database supports mTLS and TLS authentication. This section covers using JDBC Thin Connections with TLS authentication without a wallet.

- JDBC Thin Driver Connection Prerequisites for TLS Connections Without a Wallet
  Applications that use JDBC Thin driver support TLS and mutual TLS (mTLS) authentication. Connecting to an Autonomous Database instance with TLS authentication requires database credentials (username and password) and provides a secure connection, but does not require that you download Oracle wallets or Java KeyStore (JKS) files.
- Using a JDBC URL TLS Connection String for JDBC Thin Driver Without a Wallet
  To connect the database using JDBC Thin Driver with TLS without a wallet, you provide a connection string. Each database service has its own TNS Name and connection string.

JDBC Thin Driver Connection Prerequisites for TLS Connections Without a Wallet

Applications that use JDBC Thin driver support TLS and mutual TLS (mTLS) authentication. Connecting to an Autonomous Database instance with TLS authentication requires database credentials (username and password) and provides a secure connection, but does not require that you download Oracle wallets or Java KeyStore (JKS) files.

> **Note:**
>
> See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for information on allowing TLS connections.

Perform the following steps before connecting to the database:

1. **Provision Autonomous Database**: Create a database and obtain your database credentials (username and password).

   See Provision Autonomous Database for more information.

2. **Verify your JDK version for security**: If you are using JDK11, JDK10, or JDK9 then you don't need to do anything for this step. If your JDK version is less than JDK8u162 then you need to download the JCE Unlimited Strength Jurisdiction Policy Files. Refer to the `README` file for installation notes. Download the JCE files from Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8 Download.
3. **Check JDBC Driver Version**: Download the latest 18.3 JDBC Thin driver (`ojdbc8.jar` and `ucp.jar`) from Oracle Database 18c (18.3) JDBC Driver & UCP Downloads. Use the latest 18.3 JDBC driver, or newer, to take advantage of recent enhancements that simplify connections and provide easy steps for configuration. You also need the additional jars: `oraclepki.jar`, `osdt_core.jar`, and `osdt_cert.jar` for use with Oracle wallets.

Using a JDBC URL TLS Connection String for JDBC Thin Driver Without a Wallet

To connect the database using JDBC Thin Driver with TLS without a wallet, you provide a connection string. Each database service has its own TNS Name and connection string.

To run an application using the JDBC Thin Driver with TLS authentication without a wallet:

1. Copy a connection string for the Autonomous Database.

   To connect with TLS authentication copy a TLS connection string.

   a. From the Oracle Cloud Infrastructure Console, on the Autonomous Database details page click **Database connection**.

   b. Under **TLS Authentication**, select **TLS** to view the connection strings for connecting with TLS authentication.

   c. Copy a connection string.

   See View TNS Names and Connection Strings for an Autonomous Database Instance for information on viewing and copying connection stings.

   See Predefined Database Service Names for Autonomous Database for information on the different databases services for each connection string.

2. Set the `DB_URL` parameter.

   Use the following format for the `DB_URL` parameter:

   ```
   DB_URL=jdbc:oracle:thin:@my_connect_string
   ```

   For example:

   ```
   DB_URL=jdbc:oracle:thin:@(description= (retry_count=20)(retry_delay=3)
   (address=(protocol=tcps)
   (port=1521)(host=adb.region.oraclecloud.com))
   (connect_data=(service_name=u9adutfb2ba8x4d_database_medium.adb.oraclecloud
   .com))
   (security=(ssl_server_dn_match=yes)))
   ```

Preparing for Oracle Call Interface Connections

Using TLS with Oracle Call Interface connections with newer Oracle Instant Client/Oracle Database Client versions, a wallet is not required. For TLS connections with Oracle Call Interface running on older Oracle Instant Client/Oracle Database Client versions, a wallet is required for both mTLS authenticated connections and TLS authenticated connections.

- Oracle Call Interface (OCI) Connections and Wallets (mTLS)
  Oracle Net Services can find the location of the Autonomous Database wallet using the `WALLET_LOCATION` parameter in the `sqlnet.ora` file.
- Oracle Call Interface (OCI) Connections with TLS Authentication
  Using TLS with Oracle Call Interface, connections with newer Oracle Instant Client/Oracle Database Client versions, a wallet is not required.

Oracle Call Interface (OCI) Connections and Wallets (mTLS)

Oracle Net Services can find the location of the Autonomous Database wallet using the `WALLET_LOCATION` parameter in the `sqlnet.ora` file.

When `WALLET_LOCATION` is used, Oracle Net Services automatically uses the wallet. The wallet is used transparently to the application. See Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections with Wallets (mTLS) for information on setting `WALLET_LOCATION`.

See Download Client Credentials (Wallets) for information on downloading client credentials for Autonomous Database.

Oracle Call Interface (OCI) Connections with TLS Authentication

Using TLS with Oracle Call Interface, connections with newer Oracle Instant Client/Oracle Database Client versions, a wallet is not required.

Oracle Call Interface (OCI) clients support TLS authentication without a wallet if you are using the following client versions:

- Oracle Instant Client/Oracle Database Client 19.13 - only on Linux x64
- Oracle Instant Client/Oracle Database Client 19.14 (or later) and 21.5 (or later) - only on Linux x64 and Windows

> **Note:**
>
> See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for information on allowing TLS connections.

See Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections Using TLS Authentication to prepare for Oracle Call Interface connections.

When a wallet is required and you set `WALLET_LOCATION` parameter in the `sqlnet.ora` file, Oracle Net Services finds the location of the wallet and uses the wallet (use of the wallet is transparent to the application).

Predefined Database Service Names for Autonomous Database

The `tnsnames.ora` file provided with the credentials zip file contains the database service names that allow you to connect to your database.

The available predefined service names differ depending on your workload. The different services provide different levels of performance and concurrency for Autonomous Database connections.

Workload Type	Database Services Names
Data Warehouse	*dbname*_high *dbname*_medium *dbname*_low See Database Service Names for Autonomous Data Warehouse for more information.
Transaction Processing	*dbname*_tpurgent *dbname*_tp *dbname*_high *dbname*_medium *dbname*_low See Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database for more information.
JSON Database	*dbname*_tpurgent *dbname*_tp *dbname*_high *dbname*_medium *dbname*_low See Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database for more information.

Connect with Oracle Analytics Desktop

Oracle Analytics Desktop makes it easy to visualize your data so you can focus on exploring interesting data patterns. Just connect to Autonomous Database, select the elements that you're interested in, and let Oracle Analytics Desktop find the best way to visualize it. Choose from a variety of visualizations to look at data in a specific way.

For details on connecting Autonomous Database with Oracle Analytics Desktop, see *User's Guide for Oracle Analytics Desktop*.

Connect with Oracle Analytics Cloud

Oracle Analytics Cloud is a scalable and secure public cloud service that provides a full set of capabilities to explore and perform collaborative analytics for you, your workgroup, and your enterprise.

For details for connecting Autonomous Database with Oracle Analytics Cloud, see *Visualizing Data and Building Reports in Oracle Analytics Cloud*.

Connection and Networking Options and Features

Autonomous Database provides a number of different connection and networking options and features for connecting to a database.

- Using ACLs, VCNs, and Private Endpoints with Autonomous Database
  Describes the options for restricting network access to your Autonomous Database by specifying access control rules or using a virtual cloud network (private access with a private endpoint).

- Connect with Oracle Cloud Infrastructure FastConnect
  Oracle Cloud Infrastructure FastConnect provides an easy way for you to connect your on-premises network to Autonomous Database using FastConnect Public Peering. FastConnect provides higher-bandwidth options, and a more reliable and consistent networking experience compared to internet-based connections.
- Access Autonomous Database with VCN Transit Routing
  Oracle Cloud Infrastructure supports private access from your on-premises network to a database with virtual cloud network (VCN) Transit Routing.
- Access Autonomous Database with Service Gateway
  Autonomous Database supports private access from Oracle Cloud Infrastructure resources in a VCN through a service gateway.
- Use Database Resident Connection Pooling with Autonomous Database
  Database Resident Connection Pool (DRCP) in Autonomous Database supports easier and more efficient management of open connections. Using DRCP provides you with access to a connection pool in your database that enables a significant reduction in key database resources required to support many client connections and when the database needs to scale for many simultaneous connections.
- Best Practices for Low Latency Connections with Autonomous Database
  Taking steps to reduce the latency for connections between an application and Autonomous Database is critical if your application performs many round-trips between the application and the database.

Using ACLs, VCNs, and Private Endpoints with Autonomous Database

Describes the options for restricting network access to your Autonomous Database by specifying access control rules or using a virtual cloud network (private access with a private endpoint).

- About Network Access Options
  Provides an overview of the network access options available when you provision or clone Autonomous Database.
- Overview of Restricting Access with ACLs
  When you select the network access **Allow secure access from anywhere** option when you provision or clone an instance, you can restrict network access by defining an Access Control List (ACL). You can also add, update, or remove an ACL for an active instance.
- Overview of Private Endpoints
  You can specify that Autonomous Database uses a private endpoint inside your Virtual Cloud Network (VCN) in your tenancy. You can configure a private endpoint during provisioning or cloning your Autonomous Database, or you can switch to using private endpoints in existing databases that use public endpoints. This allows you to keep all traffic to and from your Autonomous Database off of the public internet.

About Network Access Options

Provides an overview of the network access options available when you provision or clone Autonomous Database.

> **Note:**
>
> For all of these options connections to your Autonomous Database use certificate-based authentication and Secure Sockets Layer (SSL).

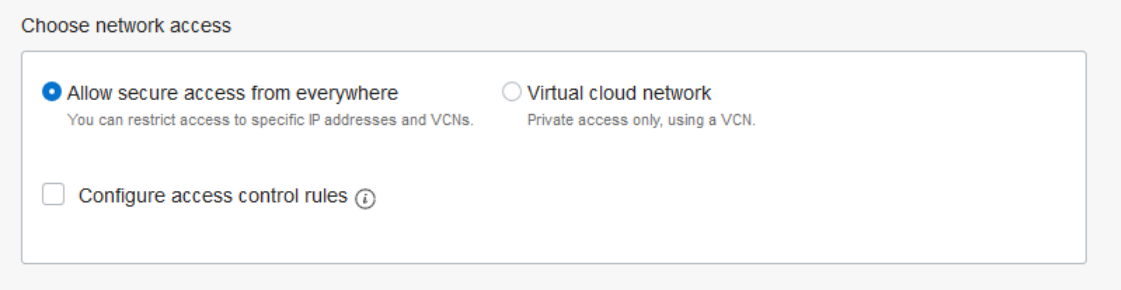
When you provision or clone your Autonomous Database you specify one of the following network access options:

- **Allow secure access from everywhere**: This option assigns a public endpoint, public IP and hostname, to your database. With this selection you have two options:
  - The database is accessible from all IP addresses: this is the default option when you provision or clone Autonomous Database.
  - Select **Configure access control rules**: This option lets you restrict access by defining access control rules in an Access Control List (ACL). By specifying an ACL, the database will be accessible from a whitelisted set of IP addresses or VCNs.

  If you configure your database with the **Allow secure access from everywhere** option, you can add, modify, or remove ACLs after you provision or clone the database.

  See Overview of Restricting Access with ACLs for more information.
- **Virtual cloud network**: This option assigns a private endpoint, private IP, and hostname to your database. Specifying this option allows traffic only from the VCN you specify; access to the database from all public IPs or VCNs is blocked. This allows you to define security rules, ingress/egress, at the Network Security Group (NSG) level and to control traffic to your Autonomous Database.

  See Configure Private Endpoints When You Provision or Clone an Instance for more information.



> **Note:**
>
> After you provision or clone your database, you can change the selection you make, to either **Allow secure access from everywhere** or **Virtual cloud network** for the database instance.

See Change from Public to Private Endpoints with Autonomous Database for more information.

See Change from Private to Public Endpoints with Autonomous Database for more information.

Overview of Restricting Access with ACLs

When you select the network access **Allow secure access from anywhere** option when you provision or clone an instance, you can restrict network access by defining an Access Control List (ACL). You can also add, update, or remove an ACL for an active instance.

Specifying an access control list blocks all IP addresses that are not in the ACL list from accessing the database. After you specify an access control list, the database only accepts connections from addresses on the access control list and the database rejects all other client connections.

Depending on where the client machines that connect to your database are located you have the following options with ACLs:

- If your client machines connect to your database through the public internet, then you can use ACLs to specify the client machine's public IP addresses or their public CIDR blocks. In this case only the specified public IP addresses can access your database.
- If the client machines reside in an Oracle Cloud Infrastructure Virtual Cloud Network (VCN), you can configure a Service Gateway to connect to your database. In this case, you can specify the VCN in your ACL, this allows all client machines in that VCN to access your database and blocks all other connections. Furthermore, you can specify the VCN and a list of private IP addresses or CIDR blocks in that VCN. This allows only those client machines with the specified IP addresses or CIDR blocks to access your database and blocks all other connections.

  See VCNs and Subnets for details on Virtual Cloud Networks (VCN).

  See Access to Oracle Services: Service Gateway for details on setting up a Service Gateway.
- If you have on-premises clients that connect to your database through Transit Routing, you can specify the VCN and also the private IP addresses or CIDR blocks of these on-premises clients to access to your database.

  See Transit Routing: Private Access to Oracle Services for details on Transit Routing.
- You can use these options together to set multiple rules to allow access from different types of clients. Multiple rules do not exclude each other.

See Configuring Network Access with Access Control Rules (ACLs) for the steps for configuring network access with ACLs, either when you provision or clone your database, or whenever you want to add, modify or remove ACLs.

Overview of Private Endpoints

You can specify that Autonomous Database uses a private endpoint inside your Virtual Cloud Network (VCN) in your tenancy. You can configure a private endpoint during provisioning or cloning your Autonomous Database, or you can switch to using private endpoints in existing databases that use public endpoints. This allows you to keep all traffic to and from your Autonomous Database off of the public internet.

Specifying the **Virtual cloud network** configuration option only allows traffic from the VCN you specify and blocks access to the database from all public IPs or VCNs. This allows you to define security rules, ingress/egress, at the Network Security Group (NSG) level and to control traffic to your database.

See Configure Network Access with Private Endpoints for the steps for configuring network access private endpoints, either when you provision or clone your database, or whenever you want to add, modify or remove private endpoints.

Connect with Oracle Cloud Infrastructure FastConnect

Oracle Cloud Infrastructure FastConnect provides an easy way for you to connect your on-premises network to Autonomous Database using FastConnect Public Peering. FastConnect

provides higher-bandwidth options, and a more reliable and consistent networking experience compared to internet-based connections.

Use FastConnect to access public services in Oracle Cloud Infrastructure without using the internet, for example, access to Object Storage, or the Oracle Cloud Infrastructure Console and APIs. Without FastConnect, the traffic destined for public IP addresses would be routed over the internet. With FastConnect, that traffic goes over your private physical connection.

For details for connecting Autonomous Database with Oracle Cloud Infrastructure FastConnect see FastConnect Overview.

Access Autonomous Database with VCN Transit Routing

Oracle Cloud Infrastructure supports private access from your on-premises network to a database with virtual cloud network (VCN) Transit Routing.

Transit routing refers to a network setup in which your on-premises network uses a connected virtual cloud network (VCN), through a service gateway on the VCN for connectivity to a database. You connect the on-premises network to the VCN with FastConnect or VPN Connect, and then configure the VCN routing so that traffic transits through the VCN to a database beyond the VCN.

After transit routing is configured, there are no special steps required to access Oracle APEX or Database Actions, or to consume RESTful services published in Oracle REST Data Services.

See Transit Routing: Private Access to Oracle Services for more details and options available with Transit Routing.

See VPN Connect for more information on VPN Connect.

Access Autonomous Database with Service Gateway

Autonomous Database supports private access from Oracle Cloud Infrastructure resources in a VCN through a service gateway.

A service gateway allows connectivity to Autonomous Database from private IP addresses in private subnets without requiring a NAT Gateway in your VCN. After a service gateway is configured, there are no special steps required to connect to Autonomous Database. Use the same connection steps as described in this chapter, depending on your application type or the client tool you are using to connect to the database.

After a service gateway is configured, there are no special steps required to access Oracle APEX or Database Actions, or to consume RESTful services published in Oracle REST Data Services.

See Access to Oracle Services: Service Gateway for details.

Use Database Resident Connection Pooling with Autonomous Database

Database Resident Connection Pool (DRCP) in Autonomous Database supports easier and more efficient management of open connections. Using DRCP provides you with access to a connection pool in your database that enables a significant reduction in key database resources required to support many client connections and when the database needs to scale for many simultaneous connections.

When you connect to Autonomous Database you choose one of the following depending on values specified in the `tnsnames.ora` configuration file:

- A dedicated server process, which services only one user process.
- A pooled server process, obtained from DRCP, which can service multiple user processes.

To connect with a pooled DRCP server process, do the following:

1. Locate or obtain the `tnsnames.ora` file you are using to connect to your Autonomous Database.

   See Download Client Credentials (Wallets) for more information.

2. Modify the `tnsnames.ora` file to add the server type `SERVER=POOLED`.

   For example:

```
example_high= (description=
     (address=(protocol=tcps)(port=1522)(host=adb.example.oraclecloud.com))
     (connect_data=(service_name=example_high.oraclecloud.com)
(SERVER=POOLED))
     (security=(ssl_server_dn_match=yes)))
```

   When you connect with `(SERVER=POOLED)` specified in the `tnsnames.ora` file you obtain a connection from DRCP.

For Autonomous Database, note the following for working with Database Resident Connection Pools (DRCP):

- DRCP is enabled by default; however using DRCP is optional. To choose a pooled connection specify `SERVER=POOLED` in `tnsnames.ora`. If you do not specify `SERVER=POOLED`, you connect with a dedicated connection.
- You cannot start or stop DRCP.

See Using Database Resident Connection Pool for more information.

Best Practices for Low Latency Connections with Autonomous Database

Taking steps to reduce the latency for connections between an application and Autonomous Database is critical if your application performs many round-trips between the application and the database.

For example, consider an OLTP application connecting to Autonomous Database and submitting thousands of SQL statements to the database individually to execute a sales order. In this case, the application requires thousands of round-trips, and reducing the latency for each round-trip can considerably speed up the sales order process. For such applications there are best practices that you can follow to reduce the latency for database connections.

- Steps to Reduce Latency for Database Connections
  You can follow these recommendations to reduce the latency for connections between your applications and the database.
- Conceptual Network Diagram for Low Latency Database Connections
  Shows the conceptual network diagram for low latency connections using public endpoints and private endpoints for your database.

Steps to Reduce Latency for Database Connections

You can follow these recommendations to reduce the latency for connections between your applications and the database.

First determine the database's availability domain. To find an Autonomous Database instance's availability domain, connect as ADMIN and run the following query:

```
SELECT json_value(cloud_identity, '$.AVAILABILITY_DOMAIN')
AVAILABILITY_DOMAIN FROM v$pdbs;
```

For example:

```
SELECT json_value(cloud_identity, '$.AVAILABILITY_DOMAIN') AVAILABILITY_DOMAIN
             FROM v$pdbs;

AVAILABILITY_DOMAIN
--------------------
SoSC:US-ASHBURN-AD-1
```

To reduce latency, do the following:

1. Place clients or the mid-tier servers in the same availability domain as the Autonomous Database instance.

   For example, if your application runs on an Oracle Cloud Infrastructure Compute VM, select the same availability domain as the Autonomous Database instance when you create the compute instance.

   If the application runs in another cloud or in an on-premises data center, use OCI FastConnect to reduce the latency for the connection to your OCI region. See FastConnect Overview for more information.

2. Configure the network routing.

   - If you are using an Autonomous Database instance on a public endpoint, configure your network routing so that the connection from the client to the database goes through a Service Gateway.

     See the following for more information.

     – Setting Up a Service Gateway in the Console

     – Access to Oracle Services: Service Gateway

   - If you are using an Autonomous Database instance on a private endpoint, you connect to the database using the private endpoint visible in your network, without the need to configure a Service Gateway.

3. Use one-way TLS connections without a wallet.

   As a best practice for lower latency, configure the Autonomous Database instance to allow both mTLS and TLS connections and use TLS connections to connect your application to the database.

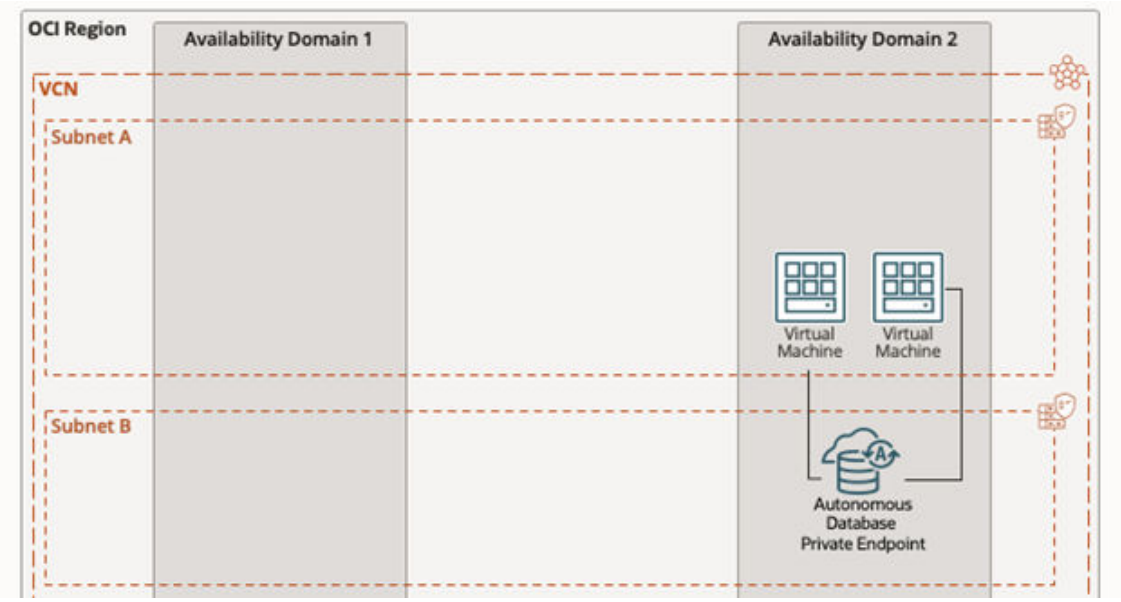
   See the following for more information:

   - Secure Connections to Autonomous Database
   - Update Network Options to Allow TLS or Require Only Mutual TLS (mTLS) Authentication on Autonomous Database

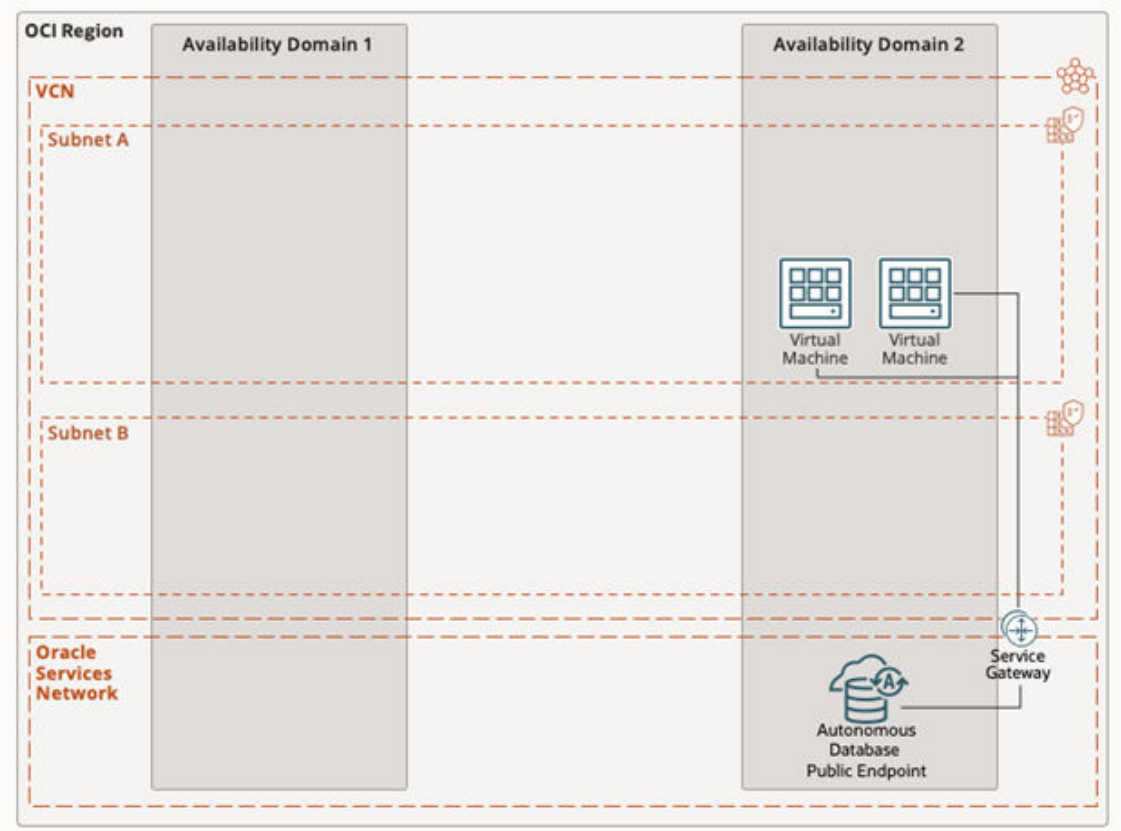
Conceptual Network Diagram for Low Latency Database Connections

Shows the conceptual network diagram for low latency connections using public endpoints and private endpoints for your database.

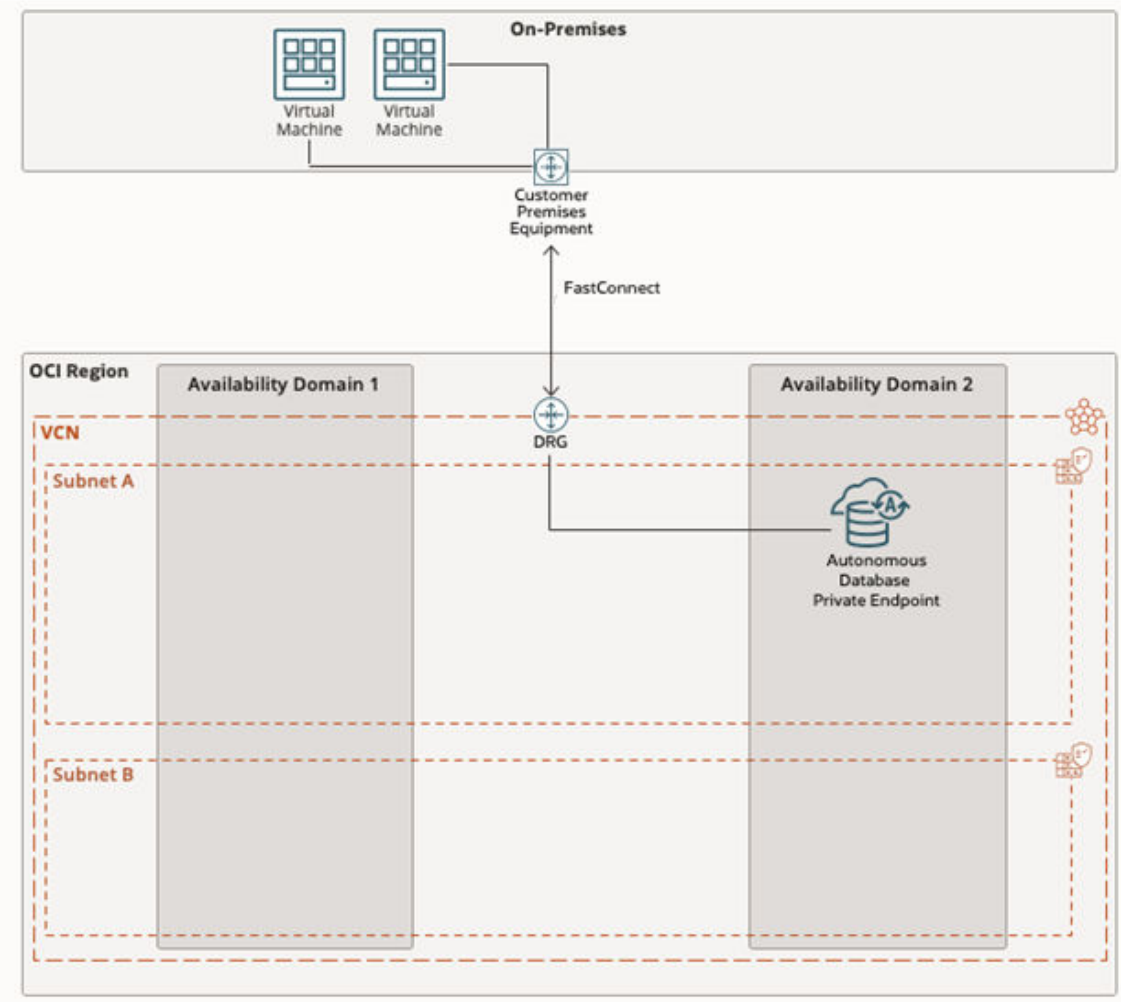
Low Latency Connections Using Private Endpoint with Application Running Inside the OCI Region



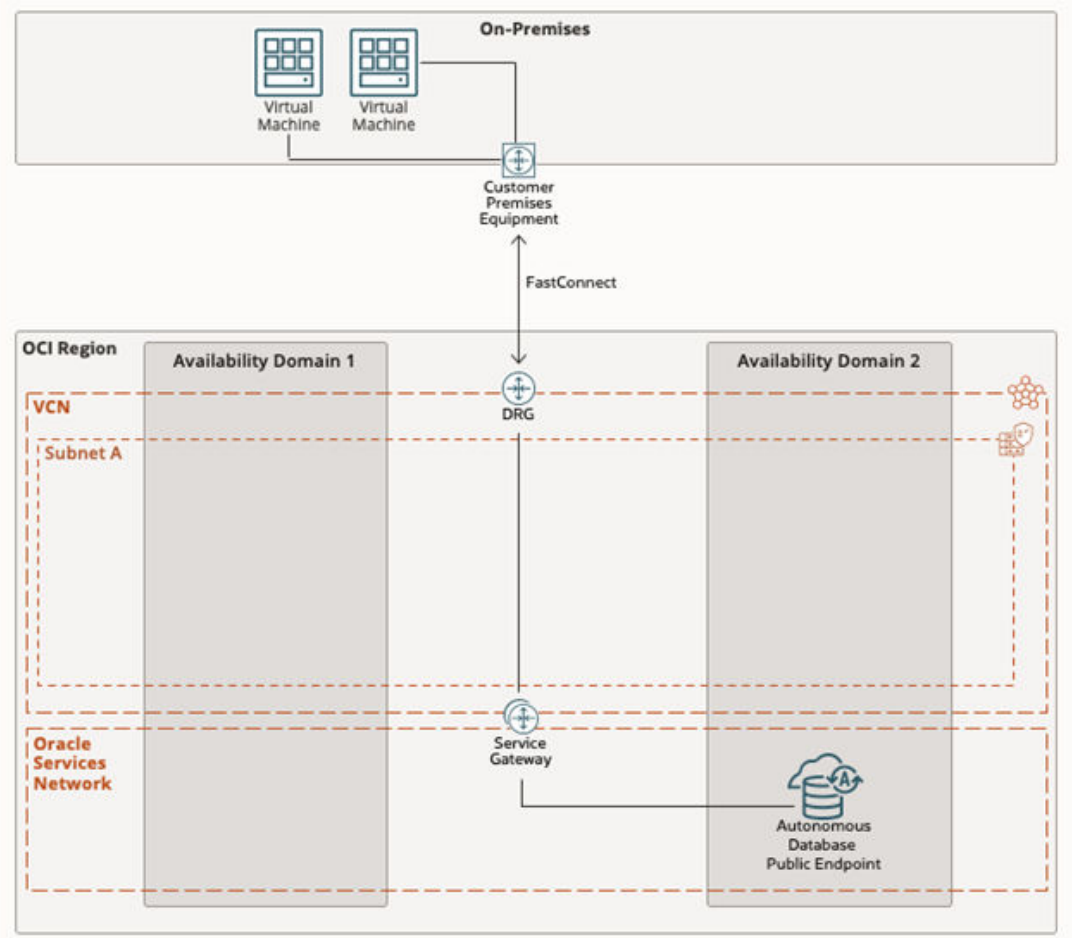
Low Latency Connections Using Public Endpoint with Application Running Inside the OCI Region



Low Latency Connections Using Private Endpoint with Application Running In On-Premies Data Center Connected to OCI Using FastConnect



Low Latency Connections Using Public Endpoint with Application Running In Your in On-Premies Data Center Connected to OCI Using FastConnect



Manage Users

Describes administration tasks for managing database users on Autonomous Database.

- Create Users on Autonomous Database
  There are several options to create users on Autonomous Database. You can use Oracle Database Actions Database Users card or use client-side tools that connect to the database to create database users.
- Remove Users on Autonomous Database
  To remove users from your database, connect to the database as the ADMIN user using any SQL client tool.
- Manage User Profiles with Autonomous Database
  You can create and alter user profiles in Autonomous Database. After you create or alter a profile, you can specify the profile clause with `CREATE USER` or `ALTER USER`. You can also import existing user profiles from another environment with Oracle Data Pump Import.
- Manage User Roles and Privileges on Autonomous Database
  There are several ways to manage user privileges and roles on Autonomous Database. You can use Oracle Database Actions Database Users card or client-side tools to connect to the database to mange privileges and roles.

- Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database
  An administrator can add an existing database user account to use with Oracle Machine Learning components or create a new user account and user credentials with the Oracle Machine Learning User Management interface.
- Use Identity and Access Management (IAM) Authentication with Autonomous Database
- Use Azure Active Directory (Azure AD) with Autonomous Database
  You can configure an Autonomous Database instance for Azure AD users to connect using Azure OAuth2 access tokens.
- Use Microsoft Active Directory with Autonomous Database
- Configure Kerberos Authentication with Autonomous Database
  Describes how to configure Kerberos to authenticate Oracle Autonomous Database users.

Create Users on Autonomous Database

There are several options to create users on Autonomous Database. You can use Oracle Database Actions Database Users card or use client-side tools that connect to the database to create database users.

- Create Users on Autonomous Database with Database Actions
  You can quickly create Autonomous Database users with Database Actions.
- Create Users on Autonomous Database - Connecting with a Client Tool
  You can create users by connecting to the database as the ADMIN user using any SQL client tool.
- Unlock User Accounts on Autonomous Database
  If a user account is locked, as the ADMIN user you can unlock the account.
- About User Passwords on Autonomous Database
  Autonomous Database requires strong passwords; the password you specify for a user must meet the minimum default password complexity rules.
- Manage the Administrator Account on Autonomous Database
  You can change the administrator user (ADMIN) password and when locked, unlock the administrator user account on Autonomous Database. When you use the APIs to create an Autonomous Database or to reset the ADMIN password, you can optionally use an Oracle Cloud Infrastructure Vault secret to store the password.

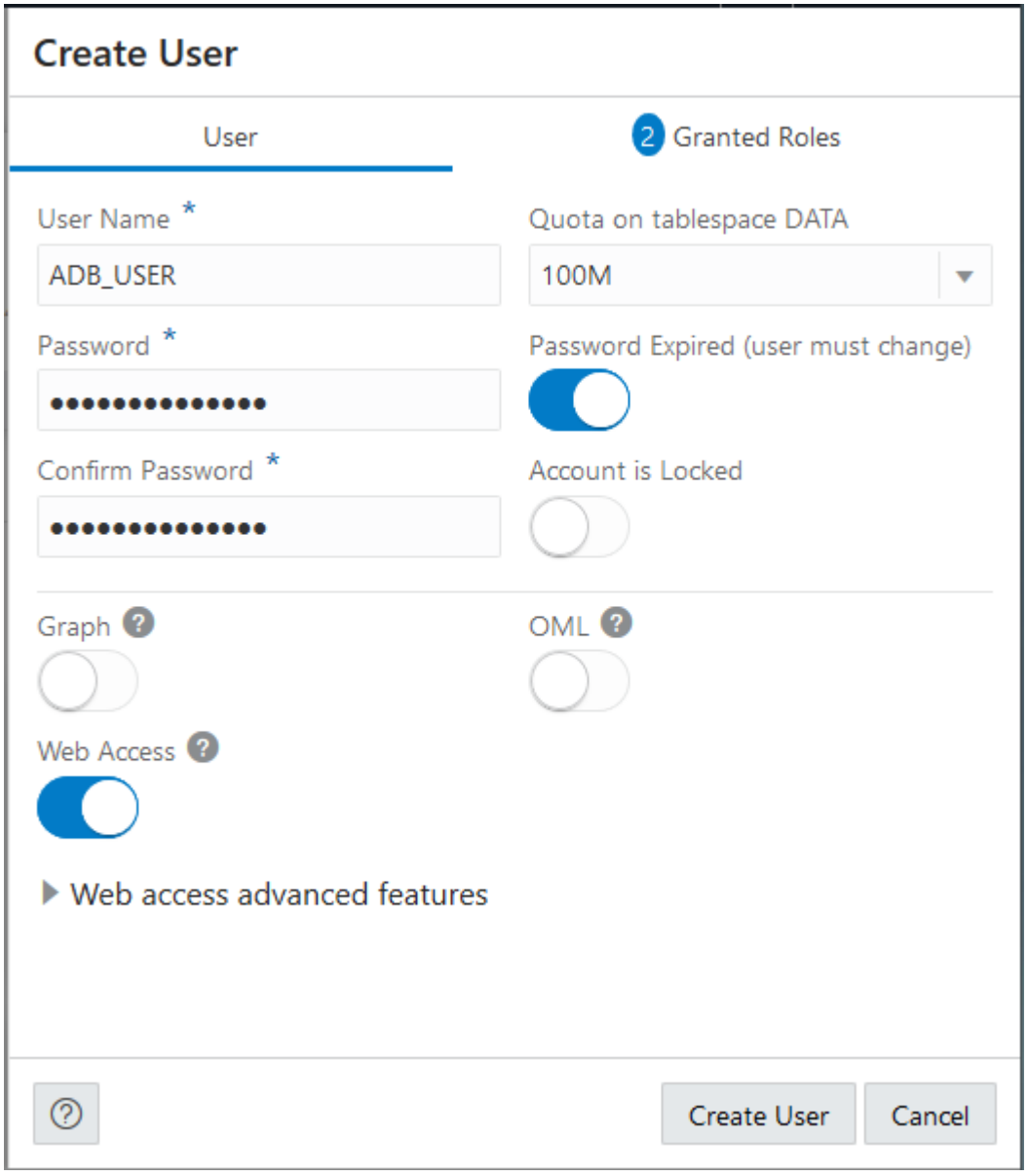
Create Users on Autonomous Database with Database Actions

You can quickly create Autonomous Database users with Database Actions.

First, access Database Actions as the ADMIN user. See Access Database Actions as ADMIN for more information.

1. Click the top left ☰ next to Oracle Database Actions.

   This shows the Database Actions menu, including **Development**, **Data Studio**, **Administration**, and **Downloads**.
2. Under **Administration** click **Database Users**.
3. On the Database Users page, in the All Users area click **Create User**.
4. To create a new user, enter a user name, a password, and enter the password again to confirm the password. Also select any options you want to enable for the user: **Graph**, **OML**, or **Web Access**.



5. Set a value for the **Quota on tablespace DATA** for the user.
6. If you want to grant roles for the new user, click the **Granted Roles** tab and select the roles for the user. For example, select DWROLE and CONNECT.
7. Click **Create User**.

   Database Actions shows the User Created confirmation message.

See Manage User Roles and Privileges on Autonomous Database for more information on granting roles and adding or updating privileges for a user.

See The Database Users Page for detailed information on Database Actions Database Users.

If you provide Web Access for the new user, then you need to send a URL to the new user. See Provide Database Actions Access to Database Users for more information.

The administrator needs to provide the credentials wallet to the new user for client-side access. See Connect to Autonomous Database for more information on client-side access credentials.

> **Note:**
>
> Autonomous Database requires strong passwords; the password you specify must meet the default password complexity rules. See About User Passwords on Autonomous Database for more information.

See Create Oracle APEX Workspaces in Autonomous Database for information on creating APEX workspaces.

See Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database to add user accounts for Oracle Machine Learning Notebooks.

Create Users on Autonomous Database - Connecting with a Client Tool

You can create users by connecting to the database as the ADMIN user using any SQL client tool.

For example, connect using Oracle SQL Developer (see Connect Oracle SQL Developer with a Wallet (mTLS)).

1. Connect as the ADMIN user.
2. Run the following SQL statements:

```
CREATE USER new_user IDENTIFIED BY password;
GRANT CREATE SESSION TO new_user;
```

> **Note:**
>
> `IDENTIFIED` with the `EXTERNALLY` clause is not supported with Autonomous Database.
> In addition, `IDENTIFIED` with the `BY VALUES` clause is not allowed.

This creates *new_user* with connect privileges. This user can now connect to the database and run queries. To grant additional privileges to users, see Manage User Roles and Privileges on Autonomous Database.

The administrator needs to provide the credentials wallet to the user *new_user*. See Connect to Autonomous Database for more information on client credentials.

> **Note:**
>
> Autonomous Database requires strong passwords; the password you specify must meet the default password complexity rules. See About User Passwords on Autonomous Database for more information.

See Provide Database Actions Access to Database Users to add users for Database Actions.

See Create Oracle APEX Workspaces in Autonomous Database for information on creating APEX workspaces.

See Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database to add user accounts for Oracle Machine Learning components.

Unlock User Accounts on Autonomous Database

If a user account is locked, as the ADMIN user you can unlock the account.

To unlock an account, connect to your database as the ADMIN user and run the following command:

```
ALTER USER username IDENTIFIED BY password ACCOUNT UNLOCK;
```

See *SQL Language Reference* for information on the `ALTER USER` command.

About User Passwords on Autonomous Database

Autonomous Database requires strong passwords; the password you specify for a user must meet the minimum default password complexity rules.

Autonomous Database sets minimum standards for passwords, and the default profile sets parameters to limit the number of failed login attempts.

- The password must be between 12 and 30 characters long and must include at least one uppercase letter, one lowercase letter, and one numeric character.

  Note, the password limit is shown as 60 characters in some help tooltip popups. Limit passwords to a maximum of 30 characters.
- The password cannot contain the *username*.
- The password cannot be one of the last four passwords used for the same *username*.
- The password cannot contain the double quote (") character.
- The password must not be the same password that is set less than 24 hours ago.

To change the password complexity rules and password parameter values you can alter the default profile or create a new profile and assign it to users. See Manage User Profiles with Autonomous Database for more information.

The following are the Autonomous Database default profile password parameter values:

Password Parameter	Description	Value
`FAILED_LOGIN_ATTEMPTS`	The maximum times a user can try to log in and fail before locking the account. This limit applies for regular database user accounts.	10
`PASSWORD_GRACE_TIME`	The number of days after the grace period begins during which a warning is issued and login is allowed.	30
`PASSWORD_LIFE_TIME`	The number of days the same password can be used for authentication.	360
`PASSWORD_LOCK_TIME`	The number of days an account will be locked after the specified number of consecutive failed login attempts.	1
`PASSWORD_REUSE_MAX`	The number of password changes required before the current password can be reused.	4
`PASSWORD_REUSE_TIME`	The number of days before which a password cannot be reused.	1

See Manage User Profiles with Autonomous Database for information on using `CREATE USER` or `ALTER USER` with a profile clause.

See *SQL Language Reference* for information on the `ALTER USER` command.

Manage the Administrator Account on Autonomous Database

You can change the administrator user (ADMIN) password and when locked, unlock the administrator user account on Autonomous Database. When you use the APIs to create an Autonomous Database or to reset the ADMIN password, you can optionally use an Oracle Cloud Infrastructure Vault secret to store the password.

See CreateAutonomousDatabase for more information.

- Set the ADMIN Password in Autonomous Database
  Provides the steps to set the ADMIN password.
- Unlock the ADMIN Account in Autonomous Database
  Shows the steps to unlock the ADMIN user account.
- Use Oracle Cloud Infrastructure Vault Secret for ADMIN Password

Set the ADMIN Password in Autonomous Database

Provides the steps to set the ADMIN password.

From the Oracle Cloud Infrastructure Console, change the password for the ADMIN user by following these steps:

1. On the **Details** page, from the **More actions** drop-down list, select **Administrator password**.
2. On the **Administrator password** page enter the new password and confirm.
3. Click **Update**.

> **Note:**
>
> You can also use Database Actions to change the password for the ADMIN user. See Manage Users and User Roles on Autonomous Database - Connecting with Database Actions for more information.

The password for the default administrator account, ADMIN, has the same password complexity rules mentioned in the section Create Users on Autonomous Database - Connecting with a Client Tool.

Unlock the ADMIN Account in Autonomous Database

Shows the steps to unlock the ADMIN user account.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

Use the following steps to unlock the ADMIN account by updating the ADMIN password:

1. On the **Details** page, from the **More actions** drop-down list, select **Administrator password**.
2. On the **Administrator password** page enter the new password and confirm.
3. Click **Update**.

This operation unlocks the ADMIN account if it was locked.

The password for the default administrator account, ADMIN, has the same password complexity rules mentioned in the section Create Users on Autonomous Database - Connecting with a Client Tool.

Use Oracle Cloud Infrastructure Vault Secret for ADMIN Password

When you create or clone an Autonomous Database instance or when you reset the ADMIN password, you can use an Oracle Cloud Infrastructure vault secret to specify the ADMIN password.

Autonomous Database allows you to use the APIs to provide a protected vault secret as the ADMIN password, with secure access to the vault secret granted through Oracle Cloud Infrastructure IAM policies.

> **Note:**
>
> Using an Oracle Cloud Infrastructure vault secret for the ADMIN password is only supported with the APIs.

Oracle Cloud Infrastructure Vault secrets are credentials that you use with Oracle Cloud Infrastructure services. Storing secrets in a vault provides greater security than you might achieve storing them elsewhere, such as in code or in configuration files. By calling database APIs you can use secrets from the Vault Service to set the ADMIN password. The vault secret password option is available when you create or clone an Autonomous Database instance, or when you set or reset the ADMIN password.

You create secrets using the Oracle Cloud Infrastructure Console, CLI, or API.

Notes for using a vault secret to set or reset the ADMIN password:

- In order for Autonomous Database to reach the secret in a vault, the following conditions must apply:
  - The secret must be in `current` or `previous` state.
  - If you specify a secret version in the API call, the specified secret version is used. If you do not specify a secret version, the call uses the latest secret version.
  - You must have the proper user group policy that allows `READ` access to the specific secret in a given compartment. For example:

    ```
    Allow userGroup1 to read secret-bundles in compartment training
    ```

- The password stored in the secret must conform to Autonomous Database password requirements.

See the following for more information:

- Overview of Vault
- CreateAutonomousDatabase

- UpdateAutonomousDatabaseDetails
- About User Passwords on Autonomous Database
- Managing Secrets

Remove Users on Autonomous Database

To remove users from your database, connect to the database as the ADMIN user using any SQL client tool.

For example, connect using Oracle SQL Developer.

1. Connect as the ADMIN user.
2. Run the following SQL statement:

```
DROP USER user_name CASCADE;
```

This removes *user_name* and the objects owned by that user.

> **Note:**
>
> This removes all *user_name* objects and the data owned by *user_name* is deleted.

You can also remove a database user with Oracle Database Actions. To remove a user, from the Database Users page, click the ⋮ to open the context menu for a user, and then select **Delete**. See Manage Users and User Roles on Autonomous Database - Connecting with Database Actions for more information on using Oracle Database Actions Database Users.

Manage User Profiles with Autonomous Database

You can create and alter user profiles in Autonomous Database. After you create or alter a profile, you can specify the profile clause with `CREATE USER` or `ALTER USER`. You can also import existing user profiles from another environment with Oracle Data Pump Import.

> **Note:**
>
> Autonomous Database has restrictions on the profile clause. See SQL Commands for information on `CREATE PROFILE` and `ALTER PROFILE` restrictions.

To add, modify, or remove a password parameter in a profile, including the `DEFAULT` profile you must have the `ALTER PROFILE` system privilege.

1. To add or alter a profile, as the ADMIN user run either `CREATE PROFILE` or `ALTER PROFILE`. For example:

```
CREATE PROFILE new_profile
  LIMIT PASSWORD_REUSE_MAX 10
  PASSWORD_LOCK_TIME 5;
```

If you are not the ADMIN user, then you must have `CREATE PROFILE` privilege to run `CREATE PROFILE`. If you run `ALTER PROFILE`, then you must have `ALTER PROFILE` privilege.

2. Use the new or altered profile with a `CREATE USER` or `ALTER USER` command. For example:

```
CREATE USER new_user IDENTIFIED BY password PROFILE new_profile;
GRANT CREATE SESSION TO new_user;
```

This creates *new_user* with profile *new_profile* and with connect privileges. The *new_user* can now connect to the database and run queries. To grant additional privileges to users, see Manage User Privileges on Autonomous Database - Connecting with a Client Tool.

See CREATE PROFILE for information on using `CREATE PROFILE` or `ALTER PROFILE`.

You can import existing profiles created in other environments using Oracle Data Pump Import (`impdp`). Any existing profile association with database users is preserved after importing into Autonomous Database. When a newly created user, created from an Oracle Data Pump import, attempts to login for the first time, the login is handled as follows:

- The password complexity restrictions are the same as the restrictions for any user on Autonomous Database.
- If the user's password violates the password complexity requirements, the account is expired with a 30-day grace period. In this case the user is required to change their password before the grace period ends.

> **Note:**
>
> Profile assignments for users with profile `ORA_PROTECTED_PROFILE` cannot be modified.

The following users share the `ORA_PROTECTED_PROFILE` profile and the profile assignment of these users cannot be changed:

- `ADBSNMP`
- `ADB_APP_STORE`
- `ADMIN`
- `DCAT_ADMIN`
- `GGADMIN`
- `RMAN$CATALOG`

When you create or alter a profile, you can specify a Password Verification Function (PVF) to manage password complexity. See Manage Password Complexity on Autonomous Database for more information.

- Manage Password Complexity on Autonomous Database
  You can create a Password Verify Function (PVF) and associate the PVF with a profile to manage the complexity of user passwords.
- Gradual Database Password Rollover for Applications

Manage Password Complexity on Autonomous Database

You can create a Password Verify Function (PVF) and associate the PVF with a profile to manage the complexity of user passwords.

> **Note:**
>
> The minimum password length for a user specified PVF is 8 characters and must include at least one upper case letter, one lower case letter and one numeric character. The minimum password length for the DEFAULT profile is 12 characters (the DEFAULT profile uses the `CLOUD_VERIFY_FUNCTION` PVF). The password cannot contain the username.
> Oracle recommends using a minimum password length of 12 characters. If you define a profile's PVF, and set the minimum password length to less than 12 characters, then tools such as Oracle Database Security Assessment Tool (DBSAT) and Qualys report this as a database security risk.

For example, to specify a PVF for a profile, use the following command:

```
CREATE PROFILE example_profile LIMIT PASSWORD_VERIFY_FUNCTION
ADMIN.EXAMPLE_PVF
```

If the profile is created or altered by any user other than the ADMIN user, then you must grant the `EXECUTE` privilege on the PVF. If you create a PVF and the password check fails, the database reports the `ORA-28219` error.

You can specify an Oracle supplied PVF, from one of the following:

- `CLOUD_VERIFY_FUNCTION` (this is the default password verification function for Autonomous Database):
  This function checks for the following requirements when users create or modify passwords:
  - The password must be between 12 and 30 characters long and must include at least one uppercase letter, one lowercase letter, and one numeric character.
  - The password cannot contain the username.
  - The password cannot be one of the last four passwords used for the same username.
  - The password cannot contain the double quote (") character.
  - The password must not be the same password that is set less than 24 hours ago.
- `ORA12C_STIG_VERIFY_FUNCTION`
  This function checks for the following requirements when users create or modify passwords:
  - The password has at least 15 characters.
  - The password has at least 1 lower case character and at least 1 upper case character.
  - The password has at least 1 digit.
  - The password has at least 1 special character.
  - The password differs from the previous password by at least 8 characters.

See ora12c_stig_verify_function Password Requirements for more information.

Note the following restrictions for a Password Verification Function (PVF) that you create and assign to a profile:

- If you specify a user profile, the minimum password length depends on how you define the associated PVF, as follows:
  - If a PVF is defined, then the minimum password length enforced is 8 characters with at least one upper case letter, one lower case letter and one numeric character. The password cannot contain the username.
  - If the PVF is defined as `NULL`, then the minimum password length enforced is 8 characters with at least one upper case letter, one lower case letter and one numeric character. The password cannot contain the username.
  - If the profile does not have a PVF defined, then the DEFAULT profile's PVF (`CLOUD_VERIFY_FUNCTION`) is assigned and the minimum password length enforced is 12 characters.
- If you specify a Password Verify Function (PVF) that is more strict than the default `CLOUD_VERIFY_FUNCTION`, then the new verify function is used.
- A PVF that you create must be created as a `DEFINER RIGHTS` PL/SQL function. If a `INVOKER` rights PVF is provided as input to `CREATE` or `ALTER PROFILE`, then `ORA-28220` error is thrown.
- Any PVF that you create must be created in the ADMIN user schema. If a non-ADMIN user owned PVF is provided as input to `CREATE` or `ALTER PROFILE`, then `ORA-28220` error is thrown.
- A PVF cannot be altered or dropped by a non-ADMIN user. That is, any user with the `CREATE` or `DROP ANY PROCEDURE` privilege is not allowed to alter or drop a PVF.
- If the PVF associated with a profile is dropped, then any attempt to change the password for a user who uses the PVF in their profile throws the error `ORA-7443`. Users can still login when the PVF associated with their profile is dropped. However, if a user's password is expired and the PVF is dropped, then the user cannot login.

  To recover from the `ORA-7443` error, the ADMIN user must recreate the dropped PVF and assign it to the profile, or assign an existing PVF to the profile. This allows a user change their password and login.
- The `CREATE ANY PROCEDURE` system privilege and `DROP ANY PROCEDURE` system privilege are audited for PVF security. See the `PROCEDURES` list in Listings of System and Object Privileges for more information.

See Managing the Complexity of Passwords for more information.

Gradual Database Password Rollover for Applications

An application can change its database passwords without an administrator having to schedule downtime.

To accomplish this, you can associate a profile having a non-zero limit for the `PASSWORD_ROLLOVER_TIME` password profile parameter, with an application schema. This allows the database password of the application user to be altered while allowing the older password to remain valid for the time specified by the `PASSWORD_ROLLOVER_TIME` limit. During the rollover period of time, the application instance can use either the old password or the new password to connect to the database server. When the rollover time expires, only the new password is allowed.

See Managing Gradual Database Password Rollover for Applications for more information.

Manage User Roles and Privileges on Autonomous Database

There are several ways to manage user privileges and roles on Autonomous Database. You can use Oracle Database Actions Database Users card or client-side tools to connect to the database to mange privileges and roles.

- Manage Users and User Roles on Autonomous Database - Connecting with Database Actions
  You can manage user roles for Autonomous Database users with Oracle Database Actions. The same steps also let you modify account settings for a user.
- Manage User Privileges on Autonomous Database - Connecting with a Client Tool
  Autonomous Databases come with a predefined database role named `DWROLE`. This role provides the common privileges for Autonomous Database users. Depending on the usage requirements you may also need to grant individual privileges to users.

Manage Users and User Roles on Autonomous Database - Connecting with Database Actions

You can manage user roles for Autonomous Database users with Oracle Database Actions. The same steps also let you modify account settings for a user.

First, access Database Actions as the ADMIN user. See Access Database Actions as ADMIN for more information.

1. Click the top left ☰ next to Oracle Database Actions.

   This shows the Database Actions menu, including **Development** and **Administration**.
2. Under **Administration** click **Database Users**.
3. On the Database Users page, in the card for the user you want to modify click the ⋮ to open the context menu for the user, then select **Edit**.

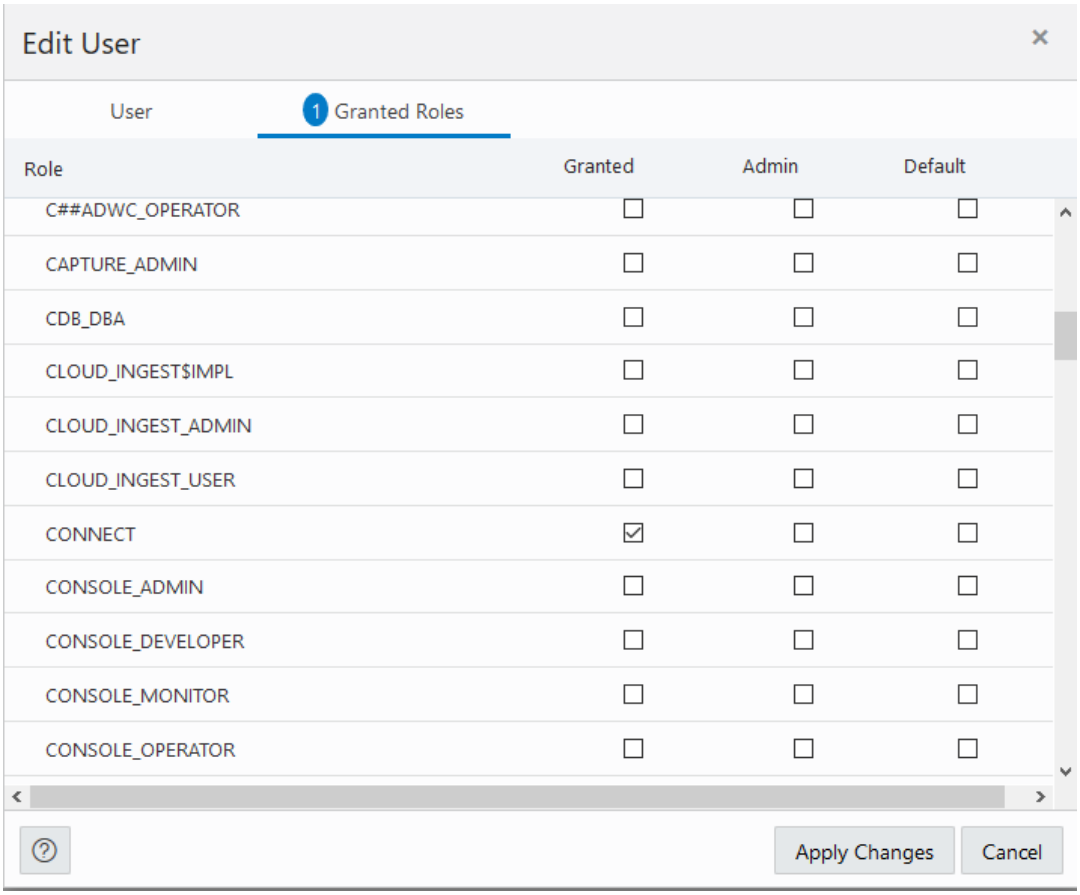
   This shows the Edit User area with the User tab selected.

   > **Note:**
   >
   > If you want to manage the user's account settings, for example if you want to provide Web Access to provide access to Database Actions, or if you want to lock the user's account, you can do this from the User tab.
4. In the Edit User area, click **Granted Roles**.

   This displays the Granted Roles tab with a list of available roles and selection boxes. For each role, you can check **Granted** to grant the role, **Admin** to permit the user to grant the role to other users, and **Default** to use the default settings for **Granted** and **Admin**.
5. Select the roles you want to grant to the user.

   For example, select CONNECT and DWROLE.

   For each role, you can select **Granted** to grant the role, **Admin** to permit the user to grant the role to other users, and **Default** to use the default settings for Granted and Admin. A new user is granted CONNECT and RESOURCE roles when **Web Access** is selected.



6. Click **Apply Changes**.

See The Database Users Page for more information on Database Actions Database Users.

See Create Users on Autonomous Database with Database Actions for information on using Database Actions.

Manage User Privileges on Autonomous Database - Connecting with a Client Tool

Autonomous Databases come with a predefined database role named `DWROLE`. This role provides the common privileges for Autonomous Database users. Depending on the usage requirements you may also need to grant individual privileges to users.

1. To grant `DWROLE` role, connect to the database as ADMIN user using any SQL client tool. For example, connect using Oracle SQL Developer (see Connect Oracle SQL Developer with a Wallet (mTLS)).
2. As the ADMIN user grant `DWROLE`. For example, the following command grants `DWROLE` to the user `adb_user`:

```
GRANT DWROLE TO adb_user;
```

3. Grant individual privileges to users with the `GRANT` command instead of or in addition to granting `DWROLE` privileges. See *Oracle Database SQL Language Reference*.
4. If a user needs to load data, do one of the following to add the privileges required to load data:

- Add quota to a new user with `CREATE USER` or alter the quota for an existing user with `ALTER USER`. For example:

```
CREATE USER sales
     QUOTA 5M on DATA;
ALTER USER sales
     QUOTA 1G on DATA;
```

- Grant `UNLIMITED TABLESPACE` privileges to a user. For example, the following command grants unlimited tablespace privileges to the user `adb_user`:

```
GRANT UNLIMITED TABLESPACE TO adb_user;
```

> **Note:**
>
> Granting `UNLIMITED TABLESPACE` privilege allows a user to use all the allocated storage space. You cannot selectively revoke tablespace access from a user with the `UNLIMITED TABLESPACE` privilege. You can grant selective or restricted access only after revoking the privilege.

The privileges in `DWROLE` are the following:

```
CREATE ANALYTIC VIEW
CREATE ATTRIBUTE DIMENSION
ALTER SESSION
CREATE HIERARCHY
CREATE JOB
CREATE MATERIALIZED VIEW
CREATE MINING MODEL
CREATE PROCEDURE
CREATE SEQUENCE
CREATE SESSION
CREATE SYNONYM
CREATE TABLE
CREATE TRIGGER
CREATE TYPE
CREATE VIEW
READ,WRITE ON directory DATA_PUMP_DIR
EXECUTE privilege on the PL/SQL package DBMS_CLOUD
EXECUTE privilege on OCI PL/SQL SDK
```

Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database

An administrator can add an existing database user account to use with Oracle Machine Learning components or create a new user account and user credentials with the Oracle Machine Learning User Management interface.

- Create User
  An administrator creates new user accounts and user credentials for Oracle Machine Learning in the User Management interface.
- Add Existing Database User Account to Oracle Machine Learning Components
  As the ADMIN user you can add an existing database user account for Oracle Machine Learning components.

Create User

An administrator creates new user accounts and user credentials for Oracle Machine Learning in the User Management interface.

> **Note:**
>
> You must have the administrator role to access the Oracle Machine Learning User Management interface.

To create a user account:

1. On the Autonomous Databases page, under the **Display Name**, select an Autonomous Database.
2. On the Autonomous Database Details page, select **Database Actions** and click **Database users**.

   As an alternative, select Database Actions and click **View all database actions** to access the Database Actions Launchpad. From the Database Actions launchpad, under **Administration** click **Database Users**.
3. Click **+ Create User**.
4. In the **User Name** field, enter a username for the account. Using the username, the user will log in to an Oracle Machine Learning instance.
5. (Optional) Select the option **Password Expired (user must change).** to required the user to change their password when they login for the first time.
6. In the **Password** field, enter a password for the user.
7. In the **Confirm Password** field, enter a password to confirm the value that you entered in the **Password** field.
8. Select **OML** to enable Oracle Machine Learning for the user.
9. Click **Create User**.

This creates a new database user and grants the required privileges to use Oracle Machine Learning.

> **Note:**
>
> With a new database user, an administrator needs to issue grant commands on the database to grant table access to the new user for the tables associated with the user's Oracle Machine Learning notebooks.

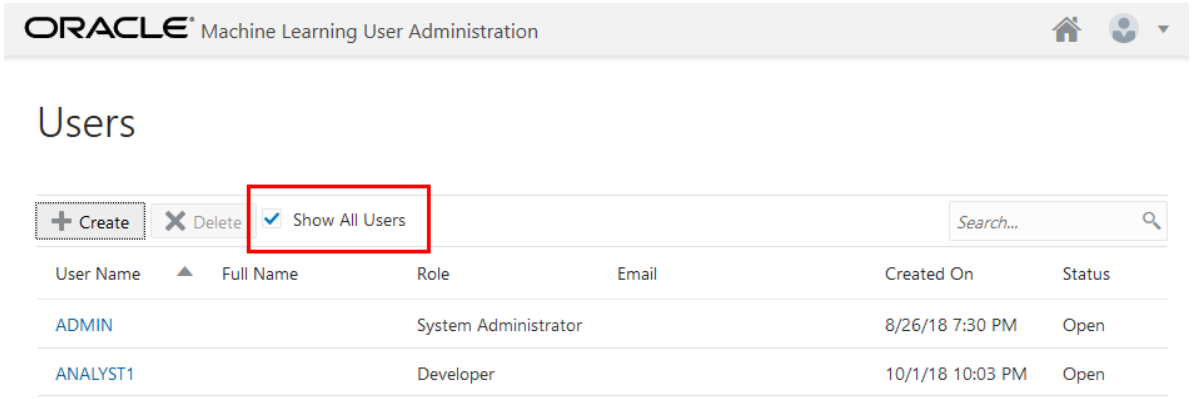
Add Existing Database User Account to Oracle Machine Learning Components

As the ADMIN user you can add an existing database user account for Oracle Machine Learning components.

> **Note:**
>
> You must have the ADMIN role to access the Oracle Machine Learning User Management interface.

To add an existing database user account:

1. On the Autonomous Databases page, under the **Display Name** column, select an Autonomous Database.
2. On the Autonomous Database Details page, select **Database Actions** and click **View all database actions**.
3. On the Database Actions Launchpad, under **Development**, click **Oracle Machine Learning**.
4. Expand the navigator by clicking the ☰ next to Oracle Machine Learning.
5. Under Admin, select **Manage OML Users** to add Oracle Machine Learning Notebooks users.
6. Click **Show All Users** to display the existing database users.



> **Note:**
>
> Initially, the **Role** field shows the role **None** for existing database users. After adding a user the role **Developer** is assigned to the user.

7. Select a user. To select a user select a name in the **User Name** column. For example, select **ANALYST1**.

   Selecting the user shows the Oracle Machine Learning **Edit User** page.
8. Enter a name in the **First Name** field. (Optional)

9. Enter the last name of the user in the **Last Name** field. (Optional)
10. In the **Email Address** field, enter the email ID of the user.

    Making any change on this page adds the existing database user with the required privileges as an Oracle Machine Learning component user.
11. Click **Save**.

This grants the required privileges to use the Oracle Machine Learning application. In Oracle Machine Learning this user can then access any tables the user has privileges to access in the database.

Use Identity and Access Management (IAM) Authentication with Autonomous Database

You can configure Autonomous Database to use Oracle Cloud Infrastructure Identity and Access Management (IAM) authentication and authorization to allow IAM users to access an Autonomous Database with IAM credentials.

> **Note:**
>
> Autonomous Database integration with Oracle Cloud Infrastructure IAM is supported in commercial regions with identity domains as well as in the legacy IAM, which does not include identity domains. IAM with identity domains was introduced with new Oracle Cloud Infrastructure tenancies that were created after November 8, 2021. Autonomous Database supports users and groups in default and non-default identity domains.

- About Identity and Access Management (IAM) Authentication with Autonomous Database
  You can enable an Autonomous Database instance to use Oracle Cloud Infrastructure (IAM) authentication and authorization for users.
- Prerequisites for Identity and Access Management (IAM) Authentication on Autonomous Database
  Describes the prerequisites for enabling IAM user access on Autonomous Database.
- Enable Identity and Access Management (IAM) Authentication on Autonomous Database
  Describes the steps to enable IAM user access on Autonomous Database.
- Create Identity and Access Management (IAM) Groups and Policies for IAM Users
  Describes the steps to write policy statements for an IAM group to enable IAM user access to Oracle Cloud Infrastructure resources, specifically Autonomous Database instances.
- Add IAM Users on Autonomous Database
  To add IAM users to allow access to Autonomous Database, map database global users to IAM groups or users with `CREATE USER` or `ALTER USER` statements (with `IDENTIFIED GLOBALLY AS` clause).
- Add IAM Roles on Autonomous Database
  Optionally, create global roles to provide additional database roles and privileges to IAM users when multiple IAM users are mapped to the same shared global user.
- Create IAM Database Password for IAM Users
  To add an IAM user and allow the IAM user to login to Autonomous Database by supplying a username and password, you must create an IAM database password.

- Connect to Autonomous Database with Identity and Access Management (IAM) Authentication
  After the ADMIN user enables Oracle Cloud Infrastructure IAM on Autonomous Database, users log in to the Autonomous Database instance using their Oracle Cloud Infrastructure IAM credentials or access the database through an Oracle Cloud Infrastructure IAM database token.
- Configure IAM Proxy Authentication
  Proxy authentication allows an IAM user to proxy to a database schema for tasks such as application maintenance.
- Disable Identity and Access Management (IAM) Authentication on Autonomous Database
  Describes the steps to disable IAM external authentication user access for Autonomous Database.
- Notes for Using Autonomous Database Tools with Identity and Access Management (IAM) Authentication
  Provides notes for using Autonomous Database tools with IAM authentication enabled.

About Identity and Access Management (IAM) Authentication with Autonomous Database

You can enable an Autonomous Database instance to use Oracle Cloud Infrastructure (IAM) authentication and authorization for users.

> **Note:**
>
> Autonomous Database integration with Oracle Cloud Infrastructure IAM is supported in commercial regions with identity domains as well as in the legacy IAM, which does not include identity domains. IAM with identity domains was introduced with new Oracle Cloud Infrastructure tenancies that were created after November 8, 2021. Autonomous Database supports users and groups in default and non-default identity domains.

Oracle Cloud Infrastructure IAM integration with Autonomous Database supports the following:

- IAM Database Password Authentication
- Identity and Access Management (IAM) SSO Token Based Authentication

See Authenticating and Authorizing IAM Users for Oracle Autonomous Databases for complete details about the architecture for using IAM users on Autonomous Database.

> **Note:**
>
> Always Free Autonomous Databases provisioned with Oracle Database 21c do not support Oracle Cloud Infrastructure Identity and Access Management (IAM) authentication and authorization.

- IAM Database Password Authentication
  You can enable an Autonomous Database instance to allow user access with an Oracle Cloud Infrastructure IAM database password (using a password verifier).

- Identity and Access Management (IAM) SSO Token Based Authentication
  You can enable an Autonomous Database instance to use Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) SSO tokens.

IAM Database Password Authentication

You can enable an Autonomous Database instance to allow user access with an Oracle Cloud Infrastructure IAM database password (using a password verifier).

> **Note:**
>
> Any supported 12c and above database client can be used for IAM database password access to Autonomous Database.

An Oracle Cloud Infrastructure IAM database password allows an IAM user to log in to an Autonomous Database instance as Oracle Database users typically log in with a user name and password. The user enters their IAM user name and IAM database password. An IAM database password is a different password than the Oracle Cloud Infrastructure Console password. Using an IAM user with the password verifier you can login to Autonomous Database with any supported database client.

For password verifier database access, you create the mappings for IAM users and OCI applications to the Autonomous Database instance. The IAM user accounts themselves are managed in IAM. The user accounts and user groups can be in either the default domain or in a custom, non-default domain.

Identity and Access Management (IAM) SSO Token Based Authentication

You can enable an Autonomous Database instance to use Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) SSO tokens.

For token verifier database access, you create the mappings for IAM users and OCI applications to the Autonomous Database instance. The IAM user accounts themselves are managed in IAM. The user accounts and user groups can be in either the default domain or in a custom, non-default domain.

There are several ways a database client can obtain an IAM database token:

- A client application or tool can request the database token from IAM for the user and can pass the database token through the client API. Using the API to send the token overrides other settings in the database client. Using IAM tokens requires the latest Oracle Database client 19c (at least 19.16). Some earlier clients (19c and 21c) provide a limited set of capabilities for token access. Oracle Database client 21c does not fully support the IAM token access feature.
- If the application or tool does not support requesting an IAM database token through the client API, the IAM user can first use Oracle Cloud Infrastructure command line interface (CLI) to retrieve the IAM database token and save it in a file location. For example, to use SQL*Plus and other applications and tools using this connection method, you first obtain the database token using the Oracle Cloud Infrastructure (OCI) Command Line Interface (CLI). If the database client is configured for IAM database tokens, when a user logs in with the slash login form, the database driver uses the IAM database token that has been saved in a default or specified file location.

- A client application or tool can use an Oracle Cloud Infrastructure IAM instance principal or resource principal to get an IAM database token, and use the IAM database token to authenticate itself to an Autonomous Database instance.
- IAM users and OCI applications can request a database token from IAM with several methods, including using an API-key. See Configuring a Client Connection for SQL*Plus That Uses an IAM Token for an example. See About Authenticating and Authorizing IAM Users for an Oracle Autonomous Database for a description of other methods such as using a delegation token within an OCI cloud shell.

Prerequisites for Identity and Access Management (IAM) Authentication on Autonomous Database

Describes the prerequisites for enabling IAM user access on Autonomous Database.

If the database is enabled for another external authentication scheme, verify that you want to use IAM on the Autonomous Database instance. There can only be one external authentication scheme enabled at any given time.

If you want to use IAM and another external authentication scheme is enabled, you can either first disable the other external authentication scheme or use the `force` parameter with the value set to true when you enable IAM authentication. See Enable Identity and Access Management (IAM) Authentication on Autonomous Database for information on using the `force` parameter.

Enable Identity and Access Management (IAM) Authentication on Autonomous Database

Describes the steps to enable IAM user access on Autonomous Database.

> **Note:**
>
> Autonomous Database integration with Oracle Cloud Infrastructure IAM is supported in commercial regions with identity domains as well as in the legacy IAM, which does not include identity domains. IAM with identity domains was introduced with new Oracle Cloud Infrastructure tenancies that were created after November 8, 2021. Autonomous Database supports users and groups in default and non-default identity domains.

To enable Autonomous Database to allow IAM users to connect to the database:

1. Perform the prerequisites for IAM authorization and authentication on Autonomous Database. See Prerequisites for Identity and Access Management (IAM) Authentication on Autonomous Database for more information.
2. Use the procedure `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION` to enable Oracle Cloud Infrastructure IAM authentication.

   When you perform these steps, connect to the Autonomous Database instance as the ADMIN user or as a user with ADMIN privileges.

   For example:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
```

```
      type => 'OCI_IAM' );
END;
/
```

By default the `force` parameter is false. When another external authentication method is enabled and `force` is false, `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION` reports the following error:

```
ORA-20004: Another external authentication is already enabled.
```

If you want to disable the external authentication that is currently enabled and use IAM authentication instead, include the `force` parameter.

For example:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
      type => 'OCI_IAM',
      force => TRUE );
END;
/
```

This sets the `IDENTITY_PROVIDER_TYPE` system parameter.

For example, you can use the following to verify `IDENTITY_PROVIDER_TYPE`:

```
SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME='identity_provider_type';

NAME                    VALUE
----------------------- -------
identity_provider_type OCI_IAM
```

Create Identity and Access Management (IAM) Groups and Policies for IAM Users

Describes the steps to write policy statements for an IAM group to enable IAM user access to Oracle Cloud Infrastructure resources, specifically Autonomous Database instances.

A policy is a group of statements that specifies who can access particular resources, and how. Access can be granted for the entire tenancy, databases in a compartment, or individual databases. This means you write a policy statement that gives a specific group a specific type of access to a specific type of resource within a specific compartment.

> **Note:**
>
> Defining a policy is required to use IAM tokens to access Autonomous Database. A policy is not required when using IAM database passwords to access Autonomous Database.

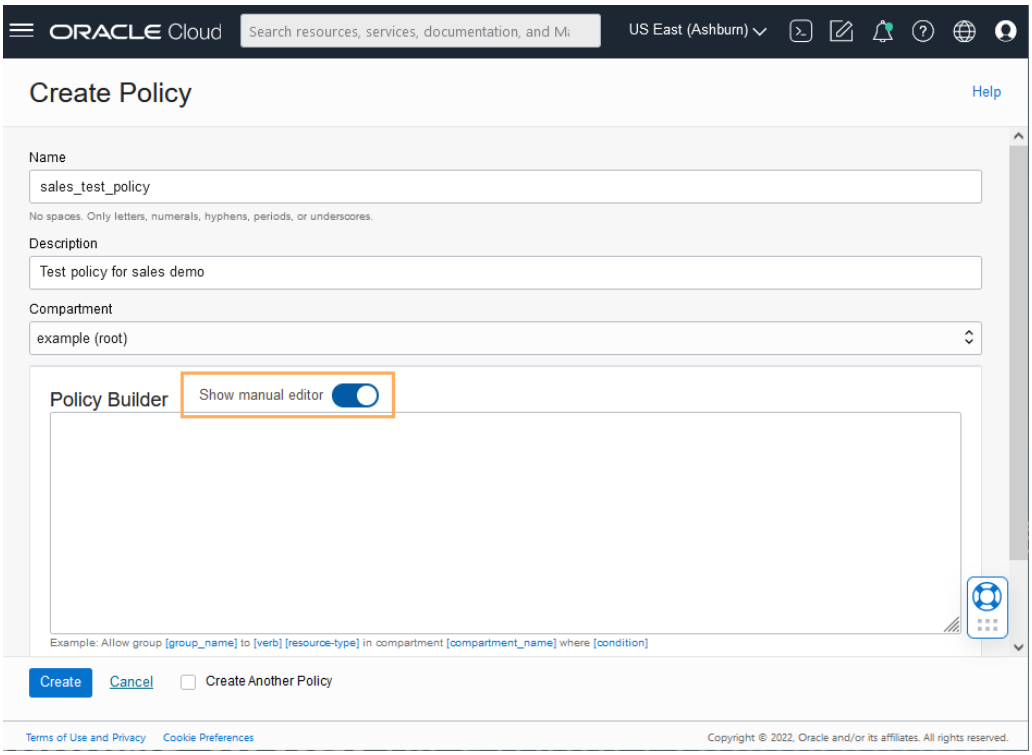
To enable Autonomous Database to allow IAM users to connect to the database using IAM tokens:

1. Perform Oracle Cloud Infrastructure Identity and Access Management prerequisites by creating a group and adding users to the group.

For example, create the group `sales_dbusers`.

See Managing Groups for more information.

2. Write policy statements to enable access to Oracle Cloud Infrastructure resources.
   a. In the Oracle Cloud Infrastructure console click **Identity & Security**.
   b. Under **Identity & Security** click **Policies**.
   c. To a write policy, click **Create Policy**.
   d. On the Create Policy page, enter a Name and a Description.
   e. On the Create Policy page, select **Show manual editor**.



   f. Use the **Policy Builder** to create a policy.

   For example to create a policy to allow users in IAM group `DBUsers` to access any Autonomous Database in their tenancy:

```
Allow group DBUsers to use autonomous-database-family in tenancy
```

   For example to create a policy that limits members of `DBUsers` group to access Autonomous Databases in compartment `testing_compartment` only:

```
allow group DBUsers to use autonomous-database-family in compartment
testing_compartment
```

For example to create a policy that limits group access to a single database in a compartment:

```
allow group DBUsers to use autonomous-database-family in compartment
testing_compartment where target.database.id =
'ocid1.autonomousdatabase.oc1.iad.aaaabbbbcccc'
```

Refer to Creating an IAM Policy to Authorize Users Authenticating with Tokens in *Database Security Guide* for more information on IAM policies to access the database.

g. Click **Create**.

See Managing Policies for more information on policies.

Notes for creating policies for use with IAM users on Autonomous Database:

- Policies can allow IAM users to access Autonomous Database instances across the entire tenancy, in a compartment, or can limit access to a single Autonomous Database instance.
- You can use either instance principal or resource principal to retrieve database tokens to establish a connection from your application to an Autonomous Database instance. If you are using an instance principal or resource principal, you must map a dynamic group. Thus, you cannot exclusively map instance and resource principals; you only can map them through a shared mapping and putting the instance or resource instance in an IAM dynamic group.

  You can create Dynamic Groups and reference dynamic groups in the policies you create to access Oracle Cloud Infrastructure. See Configure Policies and Roles to Access Resources and Managing Dynamic Groups for details.

Add IAM Users on Autonomous Database

To add IAM users to allow access to Autonomous Database, map database global users to IAM groups or users with `CREATE USER` or `ALTER USER` statements (with `IDENTIFIED GLOBALLY AS` clause).

The authorization of IAM users to an Autonomous Database instance works by mapping IAM global users (schemas) to IAM users (exclusive mapping) or IAM groups (shared schema mapping).

To authorize IAM users on an Autonomous Database instance:

1. Log in as the ADMIN user to the database that is enabled to use IAM (the ADMIN user has the required `CREATE USER` and `ALTER USER` system privileges that you need for these steps).
2. Create a mapping between the Autonomous Database user (schema) with `CREATE USER` or `ALTER USER` statements and include the `IDENTIFIED GLOBALLY AS` clause, specifying the IAM group name.

   Use the following syntax to map a global user to an IAM group:

   ```
   CREATE USER global_user IDENTIFIED GLOBALLY AS
   'IAM_GROUP_NAME=IAM_GROUP_NAME';
   ```

For example, to map an IAM group named `db_sales_group` to a shared database global user named `sales_group`:

```
CREATE USER sales_group IDENTIFIED GLOBALLY AS
     'IAM_GROUP_NAME=db_sales_group';
```

This creates a shared global user mapping. The mapping, with global user `sales_group`, is effective for all users in the IAM group. Thus, anyone in the `db_sales_group` can log in to the database using their IAM credentials (through the shared mapping of the `sales_group` global user).

The following example shows how to accomplish this for a non-default domain:

```
CREATE USER shared_sales_schema IDENTIFIED GLOBALLY AS
     'IAM_GROUP_NAME=sales_domain/db_sales_group';
```

3. If you want to create additional global user mappings for other IAM groups or users, follow these steps for each IAM group or user.

> **Note:**
>
> Database users that are not `IDENTIFIED GLOBALLY` can continue to login as before, even when the Autonomous Database is enabled for IAM authentication.

To exclusively map a local IAM user to an Oracle Database Global User:

1. Log in as the ADMIN user to the database that is enabled to use IAM (the ADMIN user has the required `CREATE USER` and `ALTER USER` system privileges that you need for these steps).
2. Create a mapping between the Autonomous Database user (schema) with `CREATE USER` or `ALTER USER` statements and include the `IDENTIFIED GLOBALLY AS` clause, specifying the IAM local IAM user name.

   For example, to create a new database global user named `peter_fitch` and map this user to an existing local IAM user named `peterfitch`:

   ```
   CREATE USER peter_fitch IDENTIFIED GLOBALLY AS
   'IAM_PRINCIPAL_NAME=peterfitch'
   ```

   The following example shows how to create the user by specifying a non-default domain, `sales_domain`:

   ```
   CREATE USER peter_fitch2 IDENTIFIED GLOBALLY AS
   'IAM_PRINCIPAL_NAME=sales_domain/peterfitch';
   ```

Add IAM Roles on Autonomous Database

Optionally, create global roles to provide additional database roles and privileges to IAM users when multiple IAM users are mapped to the same shared global user.

The use of global roles is optional when using either an exclusive IAM mapping to user (schema) or shared user mapping on Autonomous Database. For example, all privileges and

roles can be granted to the shared schema and all IAM users who map to the shared schema would be granted the privileges and roles assigned to the shared schema.

You can use a global role to optionally differentiate users who use the same shared schema. For example, a set of users can all have the same shared schema and the shared schema could have the `CREATE SESSION` privilege. Then global roles can be used to provide differentiated privileges and roles assigned to different groups of users who all use the same shared schema.

Granting additional roles to IAM users in Autonomous Database works by mapping Autonomous Database global roles to IAM groups.

To map Autonomous Database global roles to IAM groups:

1. Log in as the ADMIN user to the database that is enabled to use IAM (the ADMIN user has the required `CREATE USER` and `ALTER USER` system privileges that you need for these steps).
2. Set database authorization for Autonomous Database roles with `CREATE ROLE` or `ALTER ROLE` statements and include the `IDENTIFIED GLOBALLY AS` clause, specifying the IAM group name.

   Use the following syntax to map a global role to an IAM group:

   ```
   CREATE ROLE global_role IDENTIFIED GLOBALLY AS
       'IAM_GROUP_NAME=IAM_GROUP_of_WHICH_the_IAM_USER_IS_a_MEMBER';
   ```

   For example, to map an IAM group named `ExporterGroup` to a shared database global role named `export_role`:

   ```
   CREATE ROLE export_role IDENTIFIED GLOBALLY AS
        'IAM_GROUP_NAME=ExporterGroup';
   ```

   The following example shows how to create the role by specifying a non-default domain, `sales_domain`:

   ```
   CREATE ROLE export_role IDENTIFIED GLOBALLY AS
        'IAM_GROUP_NAME=sales_domain/ExporterGroup';
   ```

   All members of the `ExporterGroup` in the `sales_domain` domain will be authorized with the database global role `export_role` when they log in to the database.
3. Use `GRANT` statements to grant the required privileges or other roles to the global role.

   ```
   GRANT CREATE SESSION TO export_role;
   GRANT DWROLE TO export_role;
   ```

4. If you want an existing database role to be associated with an IAM group, then use `ALTER ROLE` statement to alter the existing database role to map the role to an IAM group. Use the following syntax to alter an existing database role to map it to an IAM group:

   ```
   ALTER ROLE existing_database_role
      IDENTIFIED GLOBALLY AS 'IAM_GROUP_NAME=IAM_Group_Name';
   ```

If you want to add additional global role mappings for other IAM groups, follow these steps for each IAM group.

Create IAM Database Password for IAM Users

To add an IAM user and allow the IAM user to login to Autonomous Database by supplying a username and password, you must create an IAM database password.

See Working with IAM Database Passwords for more information.

Connect to Autonomous Database with Identity and Access Management (IAM) Authentication

After the ADMIN user enables Oracle Cloud Infrastructure IAM on Autonomous Database, users log in to the Autonomous Database instance using their Oracle Cloud Infrastructure IAM credentials or access the database through an Oracle Cloud Infrastructure IAM database token.

After you enable Oracle Cloud Infrastructure IAM user access, you can also log in to the Autonomous Database using your local database account username and password (non-global database user account).

You can use a database client to access an Autonomous Database instance as an Oracle Cloud Infrastructure IAM user. To use a client with Oracle Cloud Infrastructure IAM username and password credentials and a password verifier, the database client must be 12c or newer.

Alternatively, you can use an Oracle Cloud Infrastructure IAM database token to access an Autonomous Database instance. Using IAM tokens requires the latest Oracle Database client 19c (at least 19.16). Some earlier clients (19c and 21c) provide a limited set of capabilities for token access. Oracle Database client 21c does not fully support the IAM token access feature.

The following examples show password verifier with SQL*Plus to access the database with an Oracle Cloud Infrastructure IAM username and password and the steps required to use SQL*Plus with an Oracle Cloud Infrastructure IAM database token.

> **Note:**
>
> If your Autonomous Database instance is in Restricted Mode, only the users with the `RESTRICTED SESSION` privilege such as ADMIN can connect to the database.

You can use an Oracle Cloud Infrastructure IAM database token to access an Autonomous Database instance with supported clients, including the following:

- JDBC-Thin with support for IAM Token Authentication is supported with the following:
  - JDBC version 19.13.0.0.1 (or later): See JDBC and UCP Downloads for JDBC drivers.
  - JDBC version 21.4.0.0.1 (or later): See JDBC and UCP Downloads for JDBC drivers.

  See Support for IAM Token-Based Authentication for more information:
- SQL*Plus and Oracle Instant Client: Supported with SQL*Plus and Instant Client on Linux versions 19.13 or later, and Instant Client on Linux versions 21.4 or later.
  See Identity and Access Management (IAM) Token-Based Authentication for more information.
- The database client can also be configured to retrieve a database token using the IAM username and IAM database password.

See Client Connections That Use a Token Requested by an IAM User Name and Database Password for more information.

- .NET clients (latest version of Linux or Windows). .NET software components are available as a free download from the following sites:
  - Oracle Data Access Components - .NET Downloads
  - NuGet Gallery
  - Visual Studio Code Marketplace
- About Connecting to an Autonomous Database Instance Using IAM
  IAM users can connect to the Autonomous Database instance by using either an IAM database password verifier or an IAM token.
- Configuring a Client Connection for SQL*Plus That Uses an IAM Database Password
  You can configure SQL*Plus to use an IAM database password.
- Configuring a Client Connection for SQL*Plus That Uses an IAM Token
  You can configure a client connection for SQL*Plus that uses an IAM token.
- Use Instance Principal to Access Autonomous Database with Identity and Access Management (IAM) Authentication
  After the ADMIN user enables Oracle Cloud Infrastructure IAM on Autonomous Database, an application can access the database through an Oracle Cloud Infrastructure IAM database token using an instance principal.

About Connecting to an Autonomous Database Instance Using IAM

IAM users can connect to the Autonomous Database instance by using either an IAM database password verifier or an IAM token.

Using the IAM database password verifier is similar to the Oracle Database password authentication process. However, instead of the password verifier (encrypted hash of the password) being stored in the Oracle database, the verifier is instead stored as part of the Oracle Cloud Infrastructure (OCI) IAM user profile.

The second connection method, the use of an IAM token for the database, is more modern. The use of token-based access is a better fit for Cloud resources such as Autonomous Database. The token is based on the strength that the IAM endpoint can enforce. This can be multi-factor authentication, which is stronger than the use of passwords alone. Another benefit of using tokens is that the password verifier (which is considered sensitive) is never stored or available in memory. A TCPS (TLS) connection is required when using tokens for database access.

> **Note:**
>
> You cannot configure native network encryption when passing an IAM token. Only Transport Layer Security (TLS) by itself is supported, not native network encryption or native network encryption with TLS.

- Client Connections That Use an IAM Database Password Verifier
  After you have configured the authorization needed for the IAM user, this user can log in using existing client application, such as SQL*Plus or SQLcl without additional configuration.

- Client Connections That Use a Token Requested by a Client Application or Tool
  For IAM token access to the Autonomous Database, the client application or tool requests a database token from IAM for the IAM user.

Client Connections That Use an IAM Database Password Verifier

After you have configured the authorization needed for the IAM user, this user can log in using existing client application, such as SQL*Plus or SQLcl without additional configuration.

The IAM user enters the IAM user name and IAM database password (not the Oracle Cloud Infrastructure (OCI) console password) using any currently supported database client. The only constraint is that the database client version be either Oracle Database release 12.1.0.2 or later to use Oracle Database 12c passwords. The database client must be able to use the `12C` password verifier. Using the `11G` verifier encryption is not supported with IAM. No special client or tool configuration is needed for the IAM user to connect to the OCI DBaaS instance.

Client Connections That Use a Token Requested by a Client Application or Tool

For IAM token access to the Autonomous Database, the client application or tool requests a database token from IAM for the IAM user.

The client application will pass the database token directly to the database client through the database client API.

If the application or tool has not been updated to request an IAM token, then the IAM user can use Oracle Cloud Infrastructure (OCI) command line interface (CLI) to request and store the database token. You can request a database access token (`db-token`) using the following credentials:

- Security tokens (with IAM authentication), delegation tokens (in the OCI cloud shell) and `API-keys`, which are credentials that represent the IAM user to enable the authentication
- Instance principal tokens, which enable instances to be authorized actors (or principals) to perform actions on service resources after authenticating
- Resource principal token, which is a credential that enables the application to authenticate itself to other Oracle Cloud Infrastructure services
- Using an IAM user name and IAM database password (can only be requested by database client).

When the IAM users logs into the client with a slash `/` login and the `OCI_IAM` parameter is configured (`sqlnet.ora`, `tnsnames.ora`, or as part of a connect string), then the database client retrieves the database token from a file. If the IAM user submits a user name and password, the connection will use the IAM database verifier access described for client connections that use IAM database password verifiers. The instructions in this guide show how to use the OCI CLI as a helper for the database token. If the application or tool has been updated to work with IAM, then follow the instructions for the application or tool. Some common use cases include the following: SQLPlus on-premises, SQLcl on-premises, SQL*Plus in Cloud Shell, or applications that use SEP wallets.

Configuring a Client Connection for SQL*Plus That Uses an IAM Database Password

You can configure SQL*Plus to use an IAM database password.

- As the IAM user, log in to the Autonomous Database instance by using the following syntax:

  ```
  CONNECT user_name@db_connect_string
  Enter password: password
  ```

  In this specification, *user_name* is the IAM user name. There is a limit of 128 bytes for the combined *domain_name*/*user_name*.

  The following example shows how IAM user `peter_fitch` can log in to an Autonomous Database instance.

  ```
  sqlplus /nolog
  connect peter_fitch@db_connect_string
  Enter password: password
  ```

  Some special characters will require double quotation marks around *user_name* and *password*. For example:

  ```
  "peter_fitch@example.com"@db_connect_string

  "IAM database password"
  ```

Configuring a Client Connection for SQL*Plus That Uses an IAM Token

You can configure a client connection for SQL*Plus that uses an IAM token.

1. Ensure you have an IAM user account.
2. Check with an IAM administrator and an Oracle Database administrator to ensure you have a policy allowing you to access the database in the compartment or your tenancy and that you are mapped to a global schema in the database.
3. If your application or tool does not support direct IAM integration, then download, install, and configure the OCI CLI. (See *OCI Command Line Interface Quickstart*.) Set up an API key as part of the OCI CLI configuration and select default values.
   a. Set up the API key access for the IAM user.
   b. Retrieve the `db-token`. For example:
      - Retrieving a `db-token` with an `API-key` using the Oracle Cloud Infrastructure (OCI) command-line interface:

        ```
        oci iam db-token get
        ```

      - Retrieving a `db-token` with a security (or session) token:

        ```
        oci iam db-token get --auth security_token
        ```

        If the security token has expired, a window will appear so the user can log in to OCI again. This generates the security token for the user. OCI CLI will use this refreshed token to get the `db-token`.

- Retrieving a `db-token` with a delegation token: When you log in to the cloud shell, the delegation token is automatically generated and placed in the `/etc` directory. To get this token, run the following command in the cloud shell:

  ```
  oci iam db-token get
  ```

- Retrieving an instance token by using the OCI command-line interface:

  ```
  oci iam db-token get --auth instance_principal
  ```

c. The database client can also be configured to retrieve a database token using the IAM username and IAM database password.

   See Client Connections That Use a Token Requested by an IAM User Name and Database Password for more information.

See Required Keys and OCIDs for more information.

4. Ensure that you are using the latest release updates for the Oracle Database client releases 19c, 21c, or 23ai.

   This configuration only works with the Oracle Database client release 19c, 21c, or 23ai.

5. Follow the existing process to download the wallet from the Autonomous Database and then follow the directions for configuring it for use with SQL*Plus.

   a. Confirm that DN matching is enabled by looking for `SSL_SERVER_DN_MATCH=ON` in `sqlnet.ora`.

   > **Note:**
   >
   > Partial or full DN matching is required when sending a token from the database client to Autonomous Database. If Autonomous Database is using a private endpoint, you need to specify a host value for the connect string parameter. Using an IP address for the host parameter in the connect string will not work with DN matching and the IAM token will not be sent to the database.
   > See Private Endpoints Configuration Examples on Autonomous Database for configuration information on how to set the host parameter when using a private endpoint.

   b. Configure the database client to use the IAM token by adding `TOKEN_AUTH=OCI_TOKEN` to the `sqlnet.ora` file. Because you will be using the default locations for the database token file, you do not need to include the token location.

The `TOKEN_AUTH` and `TOKEN_LOCATION` values in the `tnsnames.ora` connect strings take precedence over the `sqlnet.ora` settings for that connection. For example, for the connect string, assuming that the token is in the default location (`~/.oci/db-token` for Linux):

```
(description=
  (retry_count=20)(retry_delay=3)
  (address=(protocol=tcps)(port=1522)
  (host=example.us-phoenix-1.oraclecloud.com))

(connect_data=(service_name=aaabbbccc_exampledb_high.example.oraclecloud.co
m))
```

```
    (security=(ssl_server_dn_match=yes))
    (TOKEN_AUTH=OCI_TOKEN)))
```

After the connect string is updated with the `TOKEN_AUTH` parameter, the IAM user can log in to the Autonomous Database instance by running the following command to start SQL*Plus. You can include the connect descriptor itself or use the name of the descriptor from the `tnsnames.ora` file.

```
connect /@exampledb_high
```

Or:

```
connect /@(description=
  (retry_count=20)(retry_delay=3)
  (address=(protocol=tcps)(port=1522)
  (host=example.us-phoenix-1.oraclecloud.com))

(connect_data=(service_name=aaabbbccc_exampledb_high.example.oraclecloud.com))
  (security=(ssl_server_cert_dn="CN=example.uscom-east-1.oraclecloud.com,
     O=Example Corporation,
     L=Redwood City, ST=California, C=US")
  (TOKEN_AUTH=OCI_TOKEN)))
```

The database client is already configured to get a `db-token` because `TOKEN_AUTH` has already been set, either through the `sqlnet.ora` file or in a connect string. The database client gets the `db-token` and signs it using the private key and then sends the token to the Autonomous Database. If an IAM user name and IAM database password are specified instead of slash `/`, then the database client will connect using the password instead of using the `db-token`.

Use Instance Principal to Access Autonomous Database with Identity and Access Management (IAM) Authentication

After the ADMIN user enables Oracle Cloud Infrastructure IAM on Autonomous Database, an application can access the database through an Oracle Cloud Infrastructure IAM database token using an instance principal.

See Accessing the Oracle Cloud Infrastructure API Using Instance Principals for more information.

Configure IAM Proxy Authentication

Proxy authentication allows an IAM user to proxy to a database schema for tasks such as application maintenance.

- About Configuring IAM Proxy Authentication
  IAM users can connect to Oracle DBaaS by using proxy authentication.
- Configure Proxy Authentication for the IAM User
  To configure proxy authentication for an IAM user, the IAM user must already have a mapping to a global schema (exclusive or shared mapping). A separate database schema for the IAM user to proxy to must also be available.
- Validate the IAM User Proxy Authentication
  You can validate the IAM user proxy configuration for both password and token authentication methods.

About Configuring IAM Proxy Authentication

IAM users can connect to Oracle DBaaS by using proxy authentication.

Proxy authentication is typically used to authenticate the real user and then authorize them to use a database schema with the schema privileges and roles in order to manage an application. Alternatives such as sharing the application schema password are considered insecure and unable to audit which actual user performed an action.

A use case can be in an environment in which a named IAM user who is an application database administrator can authenticate by using their credentials and then proxy to a database schema user (for example, `hrapp`). This authentication enables the IAM administrator to use the `hrapp` privileges and roles as user `hrapp` in order to perform application maintenance, yet still use their IAM credentials for authentication. An application database administrator can sign in to the database and then proxy to an application schema to manage this schema.

You can configure proxy authentication for both the password authentication and token authentication methods.

Configure Proxy Authentication for the IAM User

To configure proxy authentication for an IAM user, the IAM user must already have a mapping to a global schema (exclusive or shared mapping). A separate database schema for the IAM user to proxy to must also be available.

After you ensure that you have this type of user, alter the database user to allow the IAM user to proxy to it.

1. Log in to the Autonomous Database instance as a user who has the `ALTER USER` system privileges.
2. Grant permission for the IAM user to proxy to the local database user account.

   An IAM user cannot be referenced in the command so the proxy must be created between the database global user (mapped to the IAM user) and the target database user.

   In the following example, `hrapp` is the database schema to proxy to, and `peterfitch_schema` is the database global user exclusively mapped to user `peterfitch`.

   ```
   ALTER USER hrapp GRANT CONNECT THROUGH peterfitch_schema;
   ```

At this stage, the IAM user can log in to the database instance using the proxy. For example, to connect using a password verifier:

```
CONNECT peterfitch[hrapp]@connect_string
Enter password: password
```

To connect using a token:

```
CONNECT [hrapp]/@connect_string
```

Validate the IAM User Proxy Authentication

You can validate the IAM user proxy configuration for both password and token authentication methods.

1. Log in to the Autonomous Database instance as a user who has the `CREATE USER` and `ALTER USER` system privileges.
2. Connect as the IAM user and run the `SHOW USER` and `SELECT SYS_CONTEXT` commands.

   For example, suppose you want to check the proxy authentication of the IAM user `peterfitch` when they proxy to database user `hrapp`. Run the following queries after you proxy to the database using an IAM user. Depending on how you authenticate and access the database, you will get different values for these queries.

   - For password authentication, assuming the IAM user is in the default domain:

     ```
     CONNECT peterfitch[hrapp]/password\!@connect_string
     SHOW USER;
     --The output should be "USER is HRAPP"
     SELECT SYS_CONTEXT('USERENV','AUTHENTICATION_METHOD') FROM DUAL;
     --The output should be "PASSWORD_GLOBAL_PROXY"
     SELECT SYS_CONTEXT('USERENV','PROXY_USER') FROM DUAL;
     --The output should be "PETERFITCH_SCHEMA"
     SELECT SYS_CONTEXT('USERENV','CURRENT_USER') FROM DUAL;
     --The output should be "HRAPP"
     ```

   - For token authentication, for a user who is in a non-default domain, `sales_domain`:

     ```
     CONNECT [hrapp]/@connect_string
     SHOW USER;
     --The output should be USER is "HRAPP "
     SELECT SYS_CONTEXT('USERENV','AUTHENTICATION_METHOD') FROM DUAL;
     --The output should be "TOKEN_GLOBAL_PROXY"
     SELECT SYS_CONTEXT('USERENV','PROXY_USER') FROM DUAL;
     --The output should be "PETERFITCH_SCHEMA"
     SELECT SYS_CONTEXT('USERENV','CURRENT_USER') FROM DUAL;
     --The output should be "HRAPP"
     ```

Disable Identity and Access Management (IAM) Authentication on Autonomous Database

Describes the steps to disable IAM external authentication user access for Autonomous Database.

You can disable IAM user access on your Autonomous Database instance as follows:

1. Run the `DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION` procedure.

   For example:

   ```
   BEGIN
      DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
   END;
   /
   ```

2. If you also want to update access to IAM from the resource, in this case the Autonomous Database instance, you may need to remove or modify the IAM group and the policies you set up to allow access to IAM from the Autonomous Database instance.

Notes for Using Autonomous Database Tools with Identity and Access Management (IAM) Authentication

Provides notes for using Autonomous Database tools with IAM authentication enabled.

- Oracle APEX is not supported for IAM users with Autonomous Database. See Create Oracle APEX Workspaces in Autonomous Database for information on using regular database users with Autonomous Database.
- Database Actions is not supported for IAM users with Autonomous Database. See Provide Database Actions Access to Database Users for information on using regular database users with Autonomous Database.
- Oracle Machine Learning Notebooks and other components are not supported for IAM Authorized users with Autonomous Database. See Add Existing Database User Account to Oracle Machine Learning Components for information on using regular database users with Autonomous Database.

Use Azure Active Directory (Azure AD) with Autonomous Database

You can configure an Autonomous Database instance for Azure AD users to connect using Azure OAuth2 access tokens.

- About Integrating Oracle Autonomous Database with Microsoft Azure AD
  Oracle Autonomous Database and Microsoft Azure AD can be configured to allow users and applications to connect to the database using their Azure AD credentials.
- Enable Azure AD Authentication on Autonomous Database
  An Azure AD administrator and an Autonomous Database administrator perform steps to configure Azure AD authentication on Autonomous Database.
- Role and Schema Mapping for Azure AD Authentication on Autonomous Database
  Azure AD users are mapped to one database schema and optionally to one or more database roles. After mapping Azure AD users, user can connect to the Autonomous Database instance.
- Azure AD Client Configuration and Access for Autonomous Database
  After you configure Azure AD on your Autonomous Database and you map Azure AD users, there are numerous ways that a user can configure a client to the Autonomous Database instance using Azure AD tokens.
- Testing the Accessibility of the Azure Endpoint
  To use Autonomous Database with Azure AD, you must ensure that your Autonomous Database instance can access the Azure AD endpoint.

About Integrating Oracle Autonomous Database with Microsoft Azure AD

Oracle Autonomous Database and Microsoft Azure AD can be configured to allow users and applications to connect to the database using their Azure AD credentials.

Azure AD users and applications can log in with Azure AD Single Sign On (SSO) credentials to access the database. This is done with an Azure AD `OAuth2` access token that the user or application first requests from Azure AD. This `OAuth2` access token contains the user identity and database access information and is then sent to the database. Refer to Refer to the Microsoft article Passwordless authentication options for Azure Active Directory for information about configuring multi-factor and passwordless authentication.

You can perform this integration in the following Oracle Database environments:

- On-premises Oracle Database release 19.18 and later
- Oracle Autonomous Database Serverless
- Oracle Autonomous Database on Dedicated Exadata Infrastructure
- Oracle Base Database Service
- Oracle Exadata Cloud Service (Oracle ExaCS)

The instructions for configuring Azure AD use the term "Oracle Database" to encompass these environments.

This type of integration enables the Azure AD user to access an Oracle Autonomous Database instance. Azure AD users and applications can log in with Azure AD Single Sign On (SSO) credentials to get an Azure AD `OAuth2` access token to send to the database.

The Azure AD administrator creates and registers Oracle Autonomous Database with Azure AD. Within Azure AD, this is called an app registration, which is short for application registration. This is the digital information that Azure AD must know about the software that is using Azure AD. The Azure AD administrator also creates application (app) roles for the database app registration in Azure AD. App roles connect Azure users, groups, and applications to database schemas and roles. The Azure AD administrator assigns Azure AD users, groups, or applications to the app roles. These app roles are mapped to a database global schema or a global role or to both a schema and a role. An Azure AD user, group, or application that is assigned to an app role will be mapped to a database global schema, global role, or to both a schema and a role. An Oracle global schema can also be mapped exclusively to an Azure AD user. An Azure AD guest user (non-organization user) or an Azure AD service principal (application) can only be mapped to a database global schema through an Azure AD app role. An Oracle global role can only be mapped from an Azure app role and cannot be mapped from an Azure user.

Oracle Autonomous Database tools including Oracle APEX, Database Actions, Oracle Graph Studio, and Oracle Database API for MongoDB are not compatible with using Azure AD tokens to connect with the database.

Tools and applications that are updated to support Azure AD tokens can authenticate users directly with Azure AD and pass the database access token to the Oracle Autonomous Database instance. You can configure existing database tools such as SQL*Plus to use an Azure AD token from a file location. In these cases, Azure AD tokens can be retrieved using tools like Microsoft PowerShell or Azure CLI and put into a file location. An Azure AD `OAuth2` database access tokens are issued with an expiration time. The Oracle Database client driver will ensure that the token is in a valid format and that it has not expired before passing it to the database. The token is scoped for the database, which means that there is information in the token about the database where the token will be used. The app roles the Azure AD principal was assigned to in the database Azure AD app registration are included as part of the access token. The directory location for the Azure AD token should only have enough permission for the user to write the token file to the location and the database client to retrieve these files (for example, just read and write by the user). Because the token allows access to the database, it should be protected within the file system.

Azure AD users can request a token from Azure AD using a number of methods to open an Azure login window to enter their Azure AD credentials.

Oracle Autonomous Database accepts tokens representing the following Azure AD principals:

- Azure AD user, who is registered user in the Azure AD tenancy
- Guest user, who is registered as a guest user in the Azure AD tenancy
- Service, which is the registered application connecting to the database as itself with the client credential flow (connection pool use case)

Oracle Autonomous Database supports the following Azure AD authentication flows:

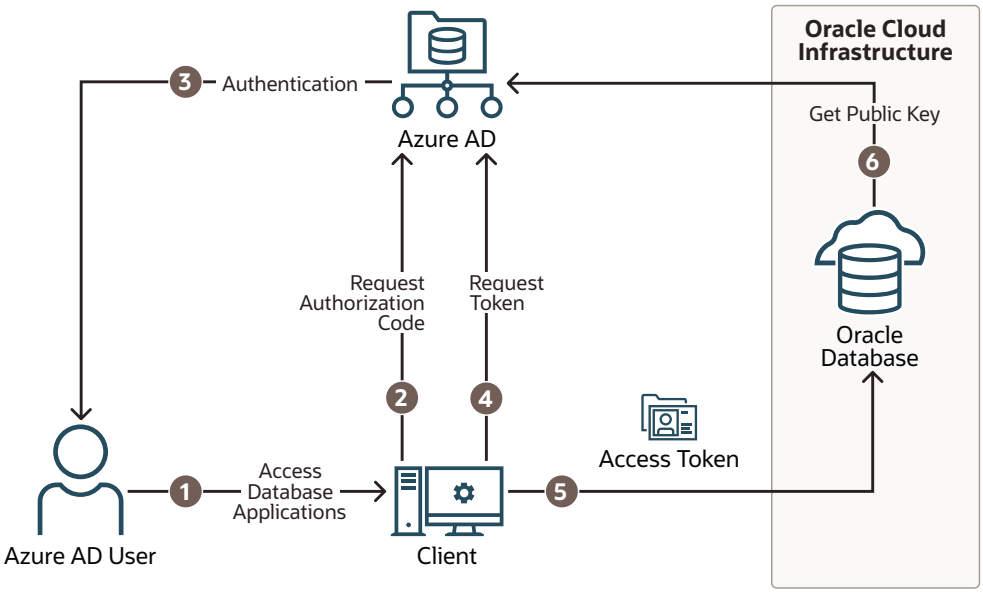
- Authorization code, most commonly used for human users (not applications) to authenticate to Azure AD in a client environment with a browser
- Client credentials, which are for database applications that connect as themselves (and not the end-user)
- On-Behalf-Of (OBO), where an application requests an access token on behalf of a logged-in user to send to the database
- Resource owner password credential (ROPC), which is not recommended for production use, but can be used in test environments where a pop-up browser user authentication would be difficult to incorporate. ROPC needs the Azure AD user name and password credential to be part of the token request call.
- Architecture of Oracle Database Integration with Microsoft Azure AD
  Microsoft Azure Active Directory access tokens follow the OAuth 2.0 standard with extensions.
- Azure AD Users Mapping to an Oracle Database Schema and Roles
  Microsoft Azure users must be mapped to an Oracle Database schema and have the necessary privileges (through roles) before being able to authenticate to the Oracle Database instance.
- Use Cases for Connecting to an Oracle Database Using Azure AD
  Oracle Database supports several use cases for connecting to the database.

Architecture of Oracle Database Integration with Microsoft Azure AD

Microsoft Azure Active Directory access tokens follow the OAuth 2.0 standard with extensions.

The Azure AD access token will be needed before you access the database from the database client (for example, with SQLPlus or SQLcl). The Oracle clients (for example, OCI, JDBC, and ODP) can be configured to pick up an Azure AD token from a file location or the token can be passed to the client through the database client API. An Azure user can use a script (examples available from Microsoft) to retrieve a token and put it into a file location for the database client to retrieve. Applications can use the Azure SDK to get an access token and pass the token through the database client API. Command-line tools such as Microsoft PowerShell or the Azure command-line interface can be used to retrieve the Azure AD token if the application cannot directly get the token.

The following diagram is a generalized flow diagram for OAuth 2.0 standard, using the `OAuth2` token. See Authentication flow support in MSAL in the Microsoft Azure AD documentation for more details about each supported flow.



The authorization code flow is an OAuth2 standard and is described in detail as part of the standards. There are two steps in the flow. The first step authenticates the user and retrieves the authorization code. The second step uses the authorization code to get the database access token.

1. The Azure AD user requests access to the resource, the Oracle Database instance.
2. The database client or application requests an authorization code from Azure AD.
3. Azure AD authenticates the Azure AD user and returns the authorization code.
4. The helper tool or application uses the authorization code with Azure AD to exchange it for the `OAuth2` token.
5. The database client sends the `OAuth2` access token to the Oracle database. The token includes the database app roles the user was assigned to in the Azure AD app registration for the database.
6. The Oracle Database instance uses the Azure AD public key to verify that the access token was created by Azure AD.

Both the database client and the database server must be registered with the **app registrations** feature in the Azure Active Directory section of the Azure portal. The database client must be registered with Azure AD app registration. Permission must also be granted to allow the database client to get an access token for the database.

Azure AD Users Mapping to an Oracle Database Schema and Roles

Microsoft Azure users must be mapped to an Oracle Database schema and have the necessary privileges (through roles) before being able to authenticate to the Oracle Database instance.

In Microsoft Azure, an Azure AD administrator can assign users, groups, and applications to the database app roles.

Exclusively mapping an Azure AD schema to a database schema requires the database administrator to create a database schema when the Azure AD user joins the organization or is authorized to the database. The database administrator must also modify the privileges and roles that are granted to the database schema to align them with the tasks the Azure AD user

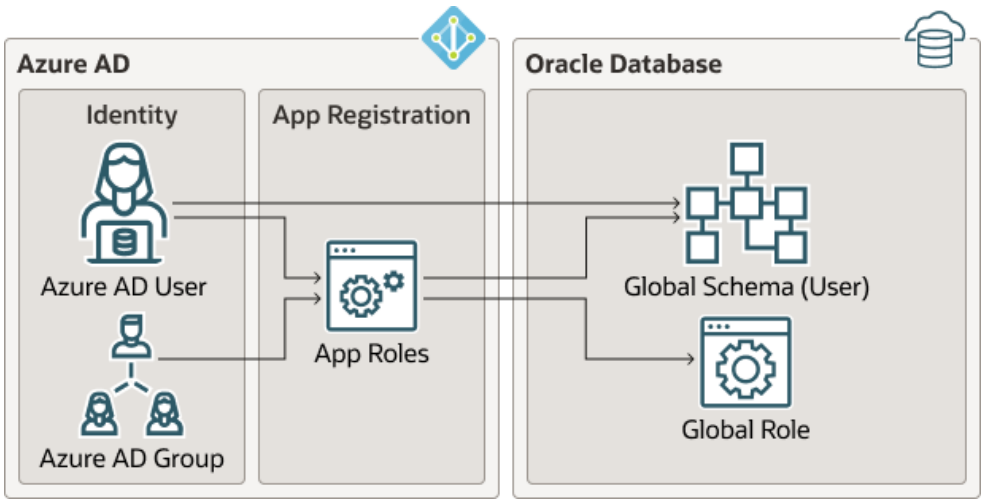
is assigned to. When the Azure AD user leaves the organization, the database administrator must drop the database schema so that an unused account is not left on the database. Using the database app roles enables the Azure AD administrator to control access and roles by assigning users to app roles that are mapped to global schemas and global roles. This way, user access to the database is managed by Azure AD administrators and database administrators do not need to create, manage, and drop schemas for every user.

An Azure AD user can be mapped to a database schema (user) either exclusively or through an app role.

- **Creating an exclusive mapping between an Azure AD user and an Oracle Database schema.** In this type of mapping, the database schema must be created for the Azure AD user. Database privileges and roles that are needed by the Azure AD user must be granted to the database schema. The database schema not only must be created when the Azure AD user is authorized to the database, but the granted privileges and roles must be modified as the Azure AD roles and tasks change. Finally, the database schema must be dropped when the Azure AD user leaves the organization.
- **Creating a shared mapping between an Azure AD app role and an Oracle Database schema.** This type of mapping, which is more common than exclusive mappings, is for Azure AD users who have been assigned directly to the app role or is a member of an Azure AD group that is assigned to the app role. The app role is mapped to an Oracle Database schema (shared schema mapping). Shared schema mapping allows multiple Azure AD users to share the same Oracle Database schema so a new database schema is not required to be created every time a new user joins the organization. This operational efficiency allows database administrators to focus on database application maintenance, performance, and tuning tasks instead of configuring new users, updating privileges and roles, and removing accounts.

In addition to database roles and privileges being granted directly to the mapped global schema, additional roles and privileges can be granted through mapped global roles. Different Azure AD users mapped to the same shared global schema may need different privileges and roles. Azure app roles can be mapped to Oracle Database global roles. Azure AD users who are assigned to the app role or are a member of an Azure AD group that is assigned to the app role will be granted the Oracle Database global role when they access the database.

The following diagram illustrates the different types of assignments and mappings that are available.



These mappings are as follows:

- An Azure AD user can be mapped directly to an Oracle Database global schema (user).
- An Azure AD user, Azure AD group, or application is assigned to an app role, which is then mapped to either an Oracle Database global schema (user) or a global role.

Use Cases for Connecting to an Oracle Database Using Azure AD

Oracle Database supports several use cases for connecting to the database.

- **OAuth2 authorization code flow:** This is the most common flow for human users. The client directs the Azure AD user to Azure AD to get the authorization code. This code is used to get the database access token. See the Microsoft Azure article Microsoft identity platform and OAuth 2.0 authorization code flow.
- **Resource owner password credentials (ROPC):** This flow is not recommended for production servers. It is useful for test software that cannot work with a pop-up authentication window. It is used in non-graphic user interface environments when a pop-up window cannot be used to authenticate a user.
- **Client credentials:** This flow is used for applications to connect with the database. The application must register with Azure AD app registration and needs a client ID and client password. These client credentials must be used to get the database access token from Azure AD when the application connects to the database. The application can pass the token through the file system or through the database client API.
- **On-behalf-of (OBO) token:** An Azure application requests an OBO token for a logged in user. The OBO token will also be an access token for the database with the Azure AD user identity and assigned app roles for the database. This enables the Azure AD user to log in to the database as the user and not the application. Only an application can request an OBO token for its Azure AD user and pass it to the database client through the API.

Enable Azure AD Authentication on Autonomous Database

An Azure AD administrator and an Autonomous Database administrator perform steps to configure Azure AD authentication on Autonomous Database.

- Registering the Oracle Database Instance with a Microsoft Azure AD Tenancy
  A user with Azure AD administrator privileges uses Microsoft Azure AD to register the Oracle Database instance with the Microsoft Azure AD tenancy.
- Enabling Microsoft Azure AD v2 Access Tokens
  To enable the Microsoft Azure AD v2 access token, you must configure it to use the `upn` attribute from the Azure portal.
- Managing App Roles in Microsoft Azure AD
  In Azure AD, you can create and manage app roles that will be assigned to Azure AD users and groups and also be mapped to Oracle Database global schemas and roles.
- Configuring Azure AD as an External Identity Provider for Autonomous Database
  An Autonomous Database administrator can enable Azure AD as an external identity provider on an Autonomous Database instance.

Enabling Microsoft Azure AD v2 Access Tokens

To enable the Microsoft Azure AD v2 access token, you must configure it to use the `upn` attribute from the Azure portal.

The Azure AD v2 access token, which is only supported on Autonomous Database Serverless, supports a wider range of access scenarios than the v1 token, including authentication for both organizational accounts (Azure AD) and personal Microsoft accounts (MSA). You can use this

token with applications that are registered in the Azure portal using the **App registrations (Preview)** experience.

1. Check the version of the Azure AD access token that you are using.
2. Log in to the Microsoft Azure portal.
3. Search for and select **Azure Active Directory**.
4. Under **Manage**, select **App registrations**.
5. Choose the application for which you want to configure optional claims based on your scenario and desired outcome.
6. Under **Manage**, select **Token configuration**.
7. Click **Add optional claim** and select **upn**.

- Checking the Azure AD Access Token Version
  You can check the version of the Microsoft Azure AD access token that your site uses by using the JSON Web Tokens web site.

Related Topics

- Checking the Azure AD Access Token Version
  You can check the version of the Microsoft Azure AD access token that your site uses by using the JSON Web Tokens web site.

Checking the Azure AD Access Token Version

You can check the version of the Microsoft Azure AD access token that your site uses by using the JSON Web Tokens web site.

By default, Azure AD Microsoft Azure AD v1 access token, but your site may have chosen to use v2. Oracle Database supports v1 tokens and Autonomous Database Serverless supports v2 tokens, as well. If you want to use the v2 access tokens, then you can enable their use for the Oracle database. To find the version of the Azure AD access token that you are using, you can either check with your Azure AD administrator, or confirm the version from the JSON Web Tokens website, as follows.

1. Go to the JSON Web Tokens website.

   ```
   https://jwt.io/
   ```

2. Copy and paste the token string into the **Encoded** field.
3. Check the **Decoded** field, which displays information about the token string.

   Near or at the bottom of the field, you will see a claim entitled `ver`, which indicates either of the following versions:

   - `"ver": "1.0"`
   - `"ver": "2.0"`

Related Topics

- Enabling Microsoft Azure AD v2 Access Tokens
  To enable the Microsoft Azure AD v2 access token, you must configure it to use the `upn` attribute from the Azure portal.

Configuring Azure AD as an External Identity Provider for Autonomous Database

An Autonomous Database administrator can enable Azure AD as an external identity provider on an Autonomous Database instance.

To enable Azure AD as an external identity provider:

1. Log in to the Autonomous Database instance as a user who has the `EXECUTE` privilege on the `DBMS_CLOUD_ADMIN` PL/SQL package. The `ADMIN` user has this privilege.
2. Run the `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION` procedure with the Azure AD required parameters.

```
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
      type   =>'AZURE_AD',
      params => JSON_OBJECT('tenant_id' VALUE 'tenant_id',
                            'application_id' VALUE 'application_id',
                            'application_id_uri' VALUE
'application_id_uri'),
      force => TRUE
  );
END;
```

In this procedure the Azure AD parameters are:

- `type`: Specifies the external authentication provider. For Azure AD, as shown, use `'AZURE_AD'`.
- `params`: Values for the required Azure AD parameters are available from the Azure portal on the app registration Overview pane for Azure Active Directory. The required `params` for Azure AD are:
  - `tenant_id`: Tenant ID of the Azure Account. Tenant Id specifies the Autonomous Database instance's Azure AD application registration.
  - `application_id`: Azure Application ID created in Azure AD to assign roles/schema mappings for external authentication in the Autonomous Database instance.
  - `application_id_uri`: Unique URI assigned to the Azure Application.
    This it the identifier for the Autonomous Database instance. The name must be domain qualified (this supports cross tenancy resource access).

    The maximum length for this parameter is 256 characters.
- `force`: Set this parameter to `TRUE` if another `EXTERNAL AUTHENTICATION` method is configured for the Autonomous Database instance and you want to disable it.

For example:

```
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
      type   =>'AZURE_AD',
      params => JSON_OBJECT('tenant_id' VALUE '29981886-6fb3-44e3-82',
                            'application_id' VALUE '11aa1a11-aaa',
                            'application_id_uri' VALUE 'https://
example.com/111aa1aa'),
      force  => TRUE
```

```
  );
END;
```

This sets the `IDENTITY_PROVIDER_TYPE` system parameter.

For example, you can use the following to verify `IDENTITY_PROVIDER_TYPE`:

```
SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME='identity_provider_type';

NAME                    VALUE
----------------------- --------
identity_provider_type AZURE_AD
```

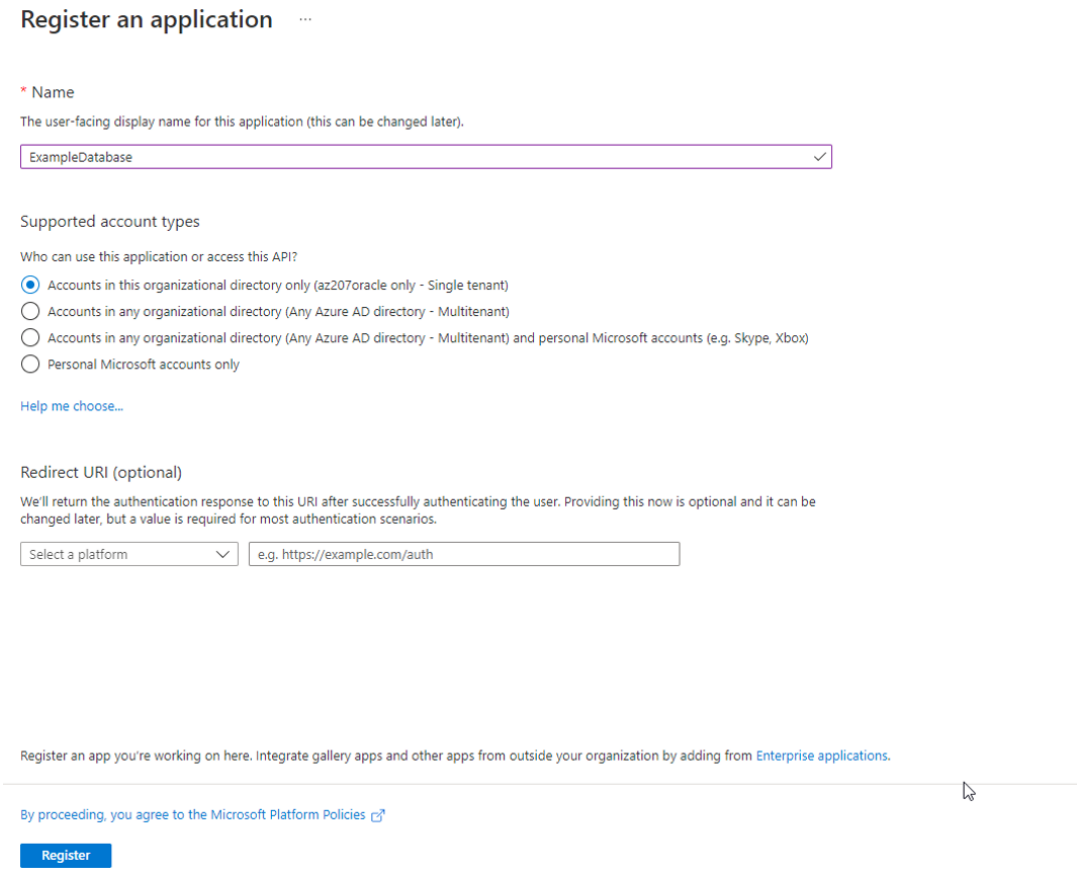
See ENABLE_EXTERNAL_AUTHENTICATION Procedure for more information.

Registering the Oracle Database Instance with a Microsoft Azure AD Tenancy

A user with Azure AD administrator privileges uses Microsoft Azure AD to register the Oracle Database instance with the Microsoft Azure AD tenancy.

1. Log in to the Azure portal as an administrator who has Microsoft Azure AD privileges to register applications.
2. In the Azure Active directory admin center page, from the left navigation bar, select **Azure Active Directory**.
3. In the MS - App registrations page, select **App registrations** from the left navigation bar.
4. Select **New registration**.

The Register an application window appears.



5. In the Register an application page, enter the following Oracle Database instance registration information:
   - In the **Name** field, enter a name for the Oracle Database instance connection (for example, *`Example Database`*).
   - Under Supported account types, select the account type that matches your use case.
     - **Accounts in this organizational directory only (*tenant_name* only - Single tenant)**
     - **Accounts in any organizational directory (Any Azure AD directory - Multitenant)**
     - **Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)**
     - **Personal Microsoft accounts only**
6. Bypass the Redirect URI (Optional) settings. You do not need to create a redirect URI because Azure AD does not need one for the database server.
7. Click **Register**.

   After you click **Register**, Azure AD displays the app registration's Overview pane, which will show the Application (client) ID under Essentials. This value is a unique identifier for the application in the Microsoft identity platform. Note the term Application refers to the Oracle Database instance.
8. Register a scope for the database app registration.

A scope is a permission to access the database. Each database will need a scope so that clients can establish a trust with the database by requesting permission to use the database scope. This allows the database client to get access tokens for the database.

a. In the left navigation bar, select **Expose an API**.

b. Under Set the App ID URI, in the **Application ID URI** field, enter the app ID URI for the database connection using the following format, and then click **Save**:

   your_tenancy_url/application_(client)_id

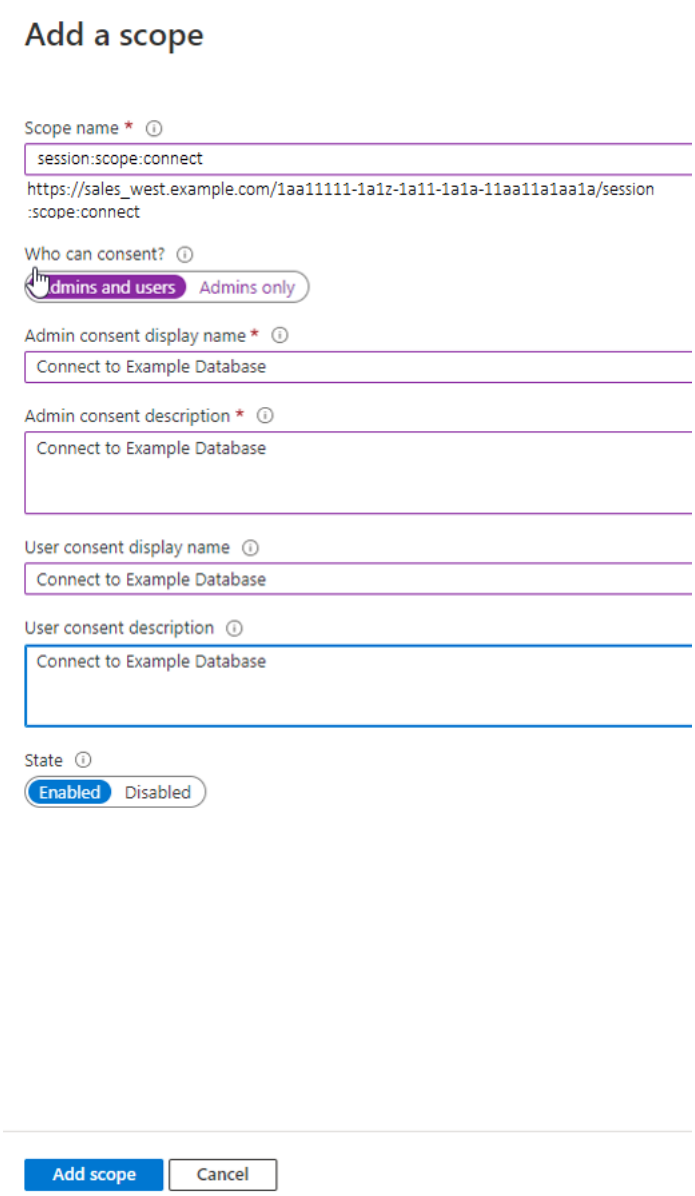
   In this specification:

   - *your_tenancy_url* must include `https` as the prefix and the fully qualified domain name of your Azure AD tenancy.
   - *application_(client)_id* is the ID that was generated when you registered the Oracle Database instance with Azure AD. It is displayed in the Overview pane of the app registration.

   For example:

   ```
   https://sales_west.example.com/1aa11111-1a1z-1a11-1a1a-11aa11a1aa1a
   ```

c. Select **Add a scope** and then enter the following settings:



- **Scope name** specifies a name for the scope. Enter the following name:

  ```
  session:scope:connect
  ```

  This name can be any text. However, a scope name must be provided. You will need to use this scope name later when you give consent to the database client application to access the database.
- **Who can consent** specifies the necessary permissions. Select **Admins and users**, or for higher restrictions, **Admins only**.
- **Admin consent** display name describes the scope's purpose (for example, `Connect to Oracle`), which only administrators can see.
- **Admin consent display name** describes the scope's purpose (for example, `Connect to Example Database`), which only administrators can see.

- **User consent display name** is a short description of the purpose of the scope (for example, `Connect to Example Database`), which users can see if you specify **Admins and users** in **Who can consent**.
- **User consent description** is a more detailed description of the purpose of the scope (for example, `Connect to Example Database`), which users can see if you specify **Admins and users** in **Who can consent**.
- **State** enables or disables the connection. Select **Enabled**.

After you complete these steps, you are ready to add one or more Azure app roles, and then perform the mappings of Oracle schemas and roles.

Related Topics

- Quickstart: Register an application with the Microsoft identity platform

Managing App Roles in Microsoft Azure AD

In Azure AD, you can create and manage app roles that will be assigned to Azure AD users and groups and also be mapped to Oracle Database global schemas and roles.

- Creating a Microsoft Azure AD App Role
  Azure AD users, groups, and applications that need to connect to the database will be assigned to the database app roles.
- Assigning Users and Groups to the Microsoft Azure AD App Role
  Before Microsoft Azure AD users can have access to the Oracle database, they must first be assigned to the app roles that will be mapped to Oracle Database schema users or roles.
- Assigning an Application to an App Role
  An application that must connect to the database using the client credential flow must to be assigned to an app role.

Creating a Microsoft Azure AD App Role

Azure AD users, groups, and applications that need to connect to the database will be assigned to the database app roles.

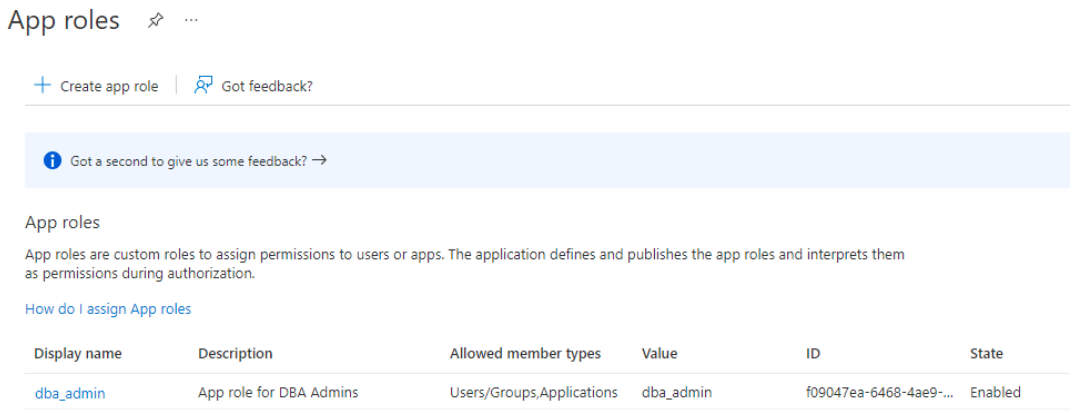
See the Microsoft Azure article Create and assign a custom role in Azure Active Directory for detailed steps on how to create an app role. The following steps describe how to create the app role for use with an Oracle database.

1. Log in to Azure AD as an administrator who has privileges for creating app roles.
2. Access the Oracle Database app registration that you created.
   a. Use the **Directory + subscription** filter to locate the Azure Active Directory tenant that contains the Oracle Database app registration.
   b. Select **Azure Active Directory**.
   c. Under **Manage**, select **App registrations**, and then select the Oracle Database instance that you registered earlier.
3. Under **Manage**, select **App roles**.
4. In the App roles page, select **Create app role**.
5. In the Create app role page, enter the following information:
   - **Display name** is the displayed name of the role (for example, `HR App Schema`). You can include spaces in this name.

- **Value** is the actual name of the role (for example, `HR_APP`). Ensure that this setting matches exactly the string that is referenced in the database mapping to a schema or role. Do not include spaces in this name.
- **Description** provides a description of the purpose of this role.
- **Do you want to enable this app role?** enables you to activate the role.

6. Click **Apply**.

   The app role appears in the App roles pane.



Assigning Users and Groups to the Microsoft Azure AD App Role

Before Microsoft Azure AD users can have access to the Oracle database, they must first be assigned to the app roles that will be mapped to Oracle Database schema users or roles.

See the Microsoft Azure article Add app roles to your application and receive them in the token for detailed steps assigning users and groups to an app role. The following steps explain how to do this for an Oracle database.

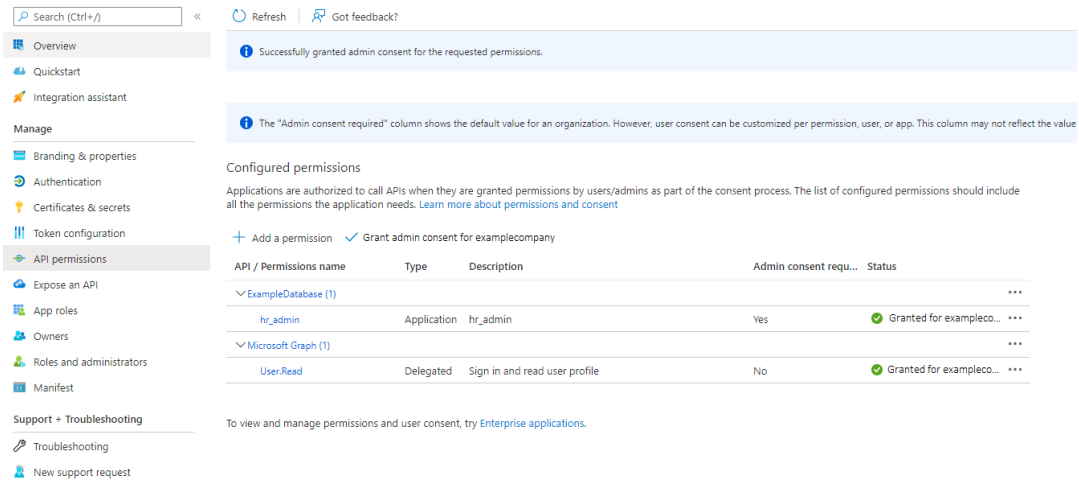
1. Log in to Azure AD as an administrator who has privileges for assigning Azure AD users and groups to app roles.
2. In enterprise applications, find the name of the Oracle Database app registration that you created. This is automatically created when you create an app registration.
   a. Use the **Directory + subscription** filter to locate the Azure Active Directory tenant that contains the Oracle connection.
   b. Select **Azure Active Directory**.
   c. Under **Manage**, select **Enterprise applications**, and then select the Oracle Database app registration name that you registered earlier.
3. Under Getting Started, select **Assign users and groups**.
4. Select **Add user/group**.
5. In the Add assignment window, select **Users and groups** to display a list of users and security groups.
6. From this list, select the users and groups that you want to assign to the app role, and then click **Select**.
7. In the Add assignment window, select **Select a role** to display a list of the app roles that you have created.

8. Select the app role and then select **Select**.
9. Click **Assign**.

Assigning an Application to an App Role

An application that must connect to the database using the client credential flow must to be assigned to an app role.

1. Log in to Azure AD as an administrator who has privileges for assigning Azure AD users and groups to app roles.
2. Access the app registration for the application.
3. Under Manage, select **API permissions**.
4. In the Configured permissions area, select **+ Add a permission**.
5. In the Request API permission pane, select the **My APIs** tab.
6. Select the Oracle Database app that you want to give permission for this application to access. Then select the **Application permissions** option.
7. Select the database app roles to assign to the application and then click the **Add Permission** box at the bottom of the screen to assign the app roles and close the dialog box. Ensure that the app roles that you just assigned appear under Configured permissions.



8. Select **Grant admin consent for *tenancy*** to grant consent for the tenancy users, then select **Yes** in the confirmation dialog box.

Related Topics

- Configure the admin consent workflow

Role and Schema Mapping for Azure AD Authentication on Autonomous Database

Azure AD users are mapped to one database schema and optionally to one or more database roles. After mapping Azure AD users, user can connect to the Autonomous Database instance.

- Exclusively Mapping an Oracle Database Schema to a Microsoft Azure AD User
  You can exclusively map an Oracle Database schema to a Microsoft Azure AD user.
- Mapping a Shared Oracle Schema to an App Role
  In this mapping, an Oracle schema is mapped to an app role. Therefore, anyone who has that app role would get the same shared schema.

- Mapping an Oracle Database Global Role to an App Role
  Oracle Database global roles that are mapped to Azure app roles give Azure users and applications additional privileges and roles above those that they have been granted through their login schemas.

Exclusively Mapping an Oracle Database Schema to a Microsoft Azure AD User

You can exclusively map an Oracle Database schema to a Microsoft Azure AD user.

1. Log in to the Oracle Database instance as a user who has been granted the `CREATE USER` or `ALTER USER` system privilege.
2. Run the `CREATE USER` or `ALTER USER` statement with the `IDENTIFIED GLOBALLY AS` clause specifying the Azure AD user name.

   For example, to create a new database schema user named `peter_fitch` and map this user to an existing Azure AD user named `peter.fitch@example.com`:

   ```
   CREATE USER peter_fitch IDENTIFIED GLOBALLY AS
   'AZURE_USER=peter.fitch@example.com';
   ```

3. Grant the `CREATE SESSION` privilege to the user.

   ```
   GRANT CREATE SESSION TO peter_fitch;
   ```

Mapping a Shared Oracle Schema to an App Role

In this mapping, an Oracle schema is mapped to an app role. Therefore, anyone who has that app role would get the same shared schema.

1. Log in to the Oracle Database instance as a user who has the `CREATE USER` or `ALTER USER` system privilege.
2. Run the `CREATE USER` or `ALTER USER` statement with the `IDENTIFIED GLOBALLY AS` clause specifying the Azure application role name.

   For example, to create a new database global user account (schema) named `dba_azure` and map it to an existing Azure AD application role named `AZURE_DBA`:

   ```
   CREATE USER dba_azure IDENTIFIED GLOBALLY AS 'AZURE_ROLE=AZURE_DBA';
   ```

Mapping an Oracle Database Global Role to an App Role

Oracle Database global roles that are mapped to Azure app roles give Azure users and applications additional privileges and roles above those that they have been granted through their login schemas.

1. Log in to the Oracle Database instance as a user who has been granted the `CREATE ROLE` or `ALTER ROLE` system privilege
2. Run the `CREATE ROLE` or `ALTER ROLE` statement with the `IDENTIFIED GLOBALLY AS` clause specifying the name of the Azure AD application role.

For example, to create a new database global role named `widget_sales_role` and map it to an existing Azure AD application role named `WidgetManagerGroup`:

```
CREATE ROLE widget_sales_role IDENTIFIED GLOBALLY AS
'AZURE_ROLE=WidgetManagerGroup';
```

Azure AD Client Configuration and Access for Autonomous Database

After you configure Azure AD on your Autonomous Database and you map Azure AD users, there are numerous ways that a user can configure a client to the Autonomous Database instance using Azure AD tokens.

If you use a wallet for your client connection (mTLS), download the wallet from the Autonomous Database instance and then follow the directions for configuring it for use with your client.

Confirm that DN matching is enabled by looking for `SSL_SERVER_DN_MATCH=ON` in `sqlnet.ora`. Partial or full DN matching is required when sending a token from the database client to Autonomous Database.

> **Note:**
>
> If Autonomous Database is using a private endpoint, you need to specify a host value for the connect string parameter. Using an IP address for the host parameter in the connect string will not work with DN matching and the Azure AD token will not be sent to the database.
> See Private Endpoints Configuration Examples on Autonomous Database for configuration information on how to set the host parameter when using a private endpoint.

See Configuring Azure AD Client Connections to the Oracle Database for more information.

Testing the Accessibility of the Azure Endpoint

To use Autonomous Database with Azure AD, you must ensure that your Autonomous Database instance can access the Azure AD endpoint.

For an Autonomous Database to accept Azure AD `OAuth2` tokens, the database must request the public key from the Azure AD endpoint.

- Run the following test to determine if the database can connect with the Azure AD endpoint:

```
SET SERVEROUTPUT ON SIZE 40000
DECLARE
  req UTL_HTTP.REQ;
  resp UTL_HTTP.RESP;
BEGIN
  UTL_HTTP.SET_WALLET(path => 'system:');
  req := UTL_HTTP.BEGIN_REQUEST('https://login.windows.net/common/
discovery/keys');
  resp := UTL_HTTP.GET_RESPONSE(req);
  DBMS_OUTPUT.PUT_LINE('HTTP response status code: ' || resp.status_code);
```

```
  UTL_HTTP.END_RESPONSE(resp);
END;
/
```

If this test is successful, then a `PL/SQL procedure successfully completed` message appears.

If the following messages appear, then it means that a database network access control list (ACL) policy blocked your test and you will need to temporarily set an access control list policy to allow you to test this:

```
ORA-29273: HTTP request failed
ORA-24247: network access denied by access control list (ACL)
```

1. Set the ACL as follows:

   ```
   BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
     host => '*',
     ace  =>  xs$ace_type(privilege_list => xs$name_list('connect'),
                          principal_name => 'username_placeholder',
                          principal_type => xs_acl.ptype_db));
   END;
   /
   ```

   Replace *username_placeholder* with the user name of the database user who is running the test. For example:

   ```
   BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
     host => '*',
     ace  =>  xs$ace_type(privilege_list => xs$name_list('connect'),
                          principal_name => 'ADB_USER',
                          principal_type => xs_acl.ptype_db));
   END;
   /
   ```

2. Try running the test again.
3. Remove the ACL, because you now no longer need it. For example, to remove the ACL for `adb_user`:

   ```
   BEGIN
   DBMS_NETWORK_ACL_ADMIN.REMOVE_HOST_ACE(
     host => '*',
     ace  =>  xs$ace_type(privilege_list => xs$name_list('connect'),
                          principal_name => 'ADB_USER',
                          principal_type => xs_acl.ptype_db));
   END;
   /
   ```

Use Microsoft Active Directory with Autonomous Database

You can configure Autonomous Database to authenticate and authorize Microsoft Active Directory users.

This configuration allows Active Directory users to access Autonomous Database using their Active Directory credentials including passwords and Kerberos.

- Prerequisites to Configure CMU with Microsoft Active Directory on Autonomous Database
- Configure CMU with Microsoft Active Directory on Autonomous Database
- Kerberos Authentication for CMU with Microsoft Active Directory
- Add Microsoft Active Directory Roles on Autonomous Database
  To add Active Directory roles, map the database global roles to Active Directory groups with `CREATE ROLE` or `ALTER ROLE` statements (and include the `IDENTIFIED GLOBALLY AS` clause).
- Add Microsoft Active Directory Users on Autonomous Database
  To add Active Directory users to access a database, map database global users to Active Directory groups or users with `CREATE USER` or `ALTER USER` statements (with `IDENTIFIED GLOBALLY AS` clause).
- Tools Restrictions with Active Directory on Autonomous Database
- Connect to Autonomous Database with Active Directory User Credentials
  After the ADMIN user completes the CMU Active Directory configuration steps and creates global roles and global users, users log in to the database using their Active Directory username and password.
- Verify Active Directory User Connection Information with Autonomous Database
  When users log in to the database using their Active Directory username and password, you can verify and audit the user activity.
- Remove Active Directory Users and Roles on Autonomous Database
  To remove Active Directory users and roles from Autonomous Databases, use standard database commands. This does not remove the related Active Directory users or groups that were mapped from the dropped database users or roles.
- Disable Active Directory Access on Autonomous Database
  Describes the steps to remove the CMU configuration from your Autonomous Database (and disable the LDAP access from your Autonomous Database to Active Directory).

Prerequisites to Configure CMU with Microsoft Active Directory on Autonomous Database

You can configure Autonomous Database to authenticate and authorize Microsoft Active Directory users.

Depending on where the Active Directory servers reside, there are two options for configuring Autonomous Database with Centrally Managed Users (CMU) with Microsoft Active Directory:

- **Active Directory (AD) servers publicly accessible**: the Active Directory servers are accessible from Autonomous Database through the public internet.
- **Active Directory (AD) servers reside on a private endpoint**: the Active Directory servers reside on a private endpoint and are not accessible from Autonomous Database through the public internet. For this case, an extra configuration step is required as shown

in last step in Configure CMU with Microsoft Active Directory on Autonomous Database where you set the database property `ROUTE_OUTBOUND_CONNECTIONS`.

> **Note:**
>
> See Use Azure Active Directory (Azure AD) with Autonomous Database for information on using Azure Active Directory with Autonomous Database. The CMU option supports Microsoft Active Directory servers but does not support the Azure Active Directory service.

The integration of Autonomous Database with Centrally Managed Users (CMU) provides integration with Microsoft Active Directory. CMU with Active Directory works by mapping Oracle database global users and global roles to Microsoft Active Directory users and groups.

The following are required prerequisites to configure the connection from Autonomous Database to Active Directory:

- You must have Microsoft Active Directory installed and configured. See AD DS Getting Started for more information.
- You must create an Oracle service directory user in Active Directory. See Connecting to Microsoft Active Directory for information on the Oracle service directory user account.
- An Active Directory system administrator must have installed Oracle password filter on the Active Directory servers, and set up Active Directory groups with Active Directory users to meet your requirements.

  > **Note:**
  >
  > This is not required if you are using Kerberos authentication for CMU Active Directory. See Kerberos Authentication for CMU with Microsoft Active Directory for more information.

  If you use password authentication with CMU Active Directory for Autonomous Database, you must use the included utility `opwdintg.exe` to install the Oracle password filter on Active Directory, extend the schema, and create three new `ORA_VFR` groups for three types of password verifier generation. See Connecting to Microsoft Active Directory for information on installing the Oracle password filter.
- You need the CMU configuration database wallet, `cwallet.sso` and the CMU configuration file `dsi.ora` to configure CMU for your Autonomous Database:
  - If you have configured CMU for an on-premise database, you can obtain these configuration files from your on-premise database server.
  - If you have not configured CMU for an on-premise database, you need to create these files. Then you upload the configuration files to the cloud to configure CMU on your Autonomous Database instance. You can validate the wallet and the `dsi.ora` by configuring CMU for an on-premise database and verifying that an Active Directory user can successfully log on to the on-premise database with these configuration files.

  For details on the wallet file for CMU, see Create the Wallet for a Secure Connection and Verify the Oracle Wallet.

  For details on the `dsi.ora` file for CMU, see Creating the dsi.ora File.

For details on configuring Active Directory for CMU and troubleshooting CMU for on-premise databases, see How To Configure Centrally Managed Users For Database Release 18c or Later Releases (Doc ID 2462012.1).

- Port 636 of the Active Directory servers must be open to Autonomous Database in Oracle Cloud Infrastructure. This allows Autonomous Database to access the Active Directory servers.
- When the Active Directory servers are on a public endpoint:
  - The Active Directory servers must be accessible from Autonomous Database through the public internet.
  - You can also extend your on-premise Active Directory to Oracle Cloud Infrastructure where you can set up Read Only Domain Controllers (RODCs) for the on-premise Active Directory. This allows you to use the RODCs in Oracle Cloud Infrastructure to authenticate and authorize on-premise Active Directory users for access to Autonomous Databases.

    See Extend Active Directory integration in Hybrid Cloud for more information.

Configure CMU with Microsoft Active Directory on Autonomous Database

You can configure Autonomous Database to authenticate and authorize Microsoft Active Directory users.

To configure Autonomous Database for CMU to connect to Active Directory:

> **Note:**
>
> When you perform the configuration steps, connect to the database as the ADMIN user.

1. Verify if another external authentication scheme is enabled on your database, and disable it.

   You can continue with CMU-AD configuration on top of Kerberos to provide CMU-AD Kerberos authentication for Microsoft Active Directory users.

   See Kerberos Authentication for CMU with Microsoft Active Directory for more information.

2. Upload the CMU configuration files, including the database wallet file, `cwallet.sso` and the CMU configuration file, `dsi.ora` to your Object Store. This step depends on the Object Store you use.

   The `dsi.ora` configuration file contains the information to find the Active Directory servers.

   If you are using Oracle Cloud Infrastructure Object Store, see Putting Data into Object Storage for details on uploading files.

3. Run `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION` procedure and pass in a location URI with the `params` JSON argument. You must place the configuration files `cwallet.sso` and `dsi.ora` in the Object Storage location specified in the `location_uri` parameter.

   For example:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
       type     => 'CMU',
```

```
       params   => JSON_OBJECT('location_uri' value 'https://
objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
                                'credential_name' value
'my_credential_name')
   );
END;
/
```

Oracle recommends that you store the CMU configuration files in a private bucket in your Object Store.

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.

The `credential_name` you use in this step is the credentials to access the Object Store.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

If the `location_uri` is a pre-authenticated URL or a pre-signed URL, then supplying a `credential_name` is not required.

The procedure creates a directory object named `CMU_WALLET_DIR` in your database and copies the CMU configuration files from the Object Store location to the directory object. This procedure also sets the database property `CMU_WALLET` to the value `'CMU_WALLET_DIR'` and sets the `LDAP_DIRECTORY_ACCESS` parameter value to the value `PASSWORD` to enable access from the Autonomous Database instance to Active Directory.

4. After you enable CMU authentication, remove the CMU configuration files including the database wallet `cwallet.sso` and the CMU configuration file `dsi.ora` from Object Store. You can use local Object Store methods to remove these files or use `DBMS_CLOUD.DELETE_OBJECT` to delete the files from Object Store.
5. When Active Directory servers are on a private endpoint, perform additional configuration steps to provide access to the private endpoint.
   a. Set the database property `ROUTE_OUTBOUND_CONNECTIONS` to the value `'PRIVATE_ENDPOINT'`.

      See Enhanced Security for Outbound Connections with Private Endpoints for more information.
   b. Validate that the CMU-AD configuration file `dsi.ora` includes host names. When `ROUTE_OUTBOUND_CONNECTIONS` is set to `'PRIVATE_TARGET'`, IP addresses cannot be specified in `dsi.ora`.

Note for CMU with Active Directory on Autonomous Database:

- Only "password authentication" and Kerberos are supported for CMU with Autonomous Database. When you are using CMU authentication with Autonomous Database, other CMU authentication methods such as PKI are not supported.

See Disable Active Directory Access on Autonomous Database for instructions to disable the access from Autonomous Database to Active Directory.

See ENABLE_EXTERNAL_AUTHENTICATION Procedure for information on `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION`.

See Configuring Centrally Managed Users with Microsoft Active Directory for more information on configuring CMU with Microsoft Active Directory.

Kerberos Authentication for CMU with Microsoft Active Directory

You can configure Autonomous Database to use Kerberos authentication for CMU with Microsoft Active Directory users. This configuration allows CMU Active Directory (CMU-AD) users to access an Autonomous Database instance using Kerberos credentials.

Kerberos can be configured with or without CMU-AD. Just configuring Kerberos requires you to create and maintain a database user for every Kerberos user. Configuring Kerberos with CMU allows you to map an Active Directory group of Kerberos users to a single database user, shared schema, so database access can be controlled by Active Directory group membership. See Configure Kerberos Authentication with Autonomous Database for details on configuring Kerberos without CMU-AD.

> **Note:**
>
> When implementing both the Kerberos authentication and CMU-AD for authorization, Oracle recommends implementing Kerberos authentication first, and then adding CMU-AD authorization.

1. Enable Kerberos in your Autonomous Database instance using Microsoft Active Directory Kerberos server.

   Only Microsoft Active Directory Kerberos servers are supported for Kerberos when you configure Kerberos authentication with CMU-AD.

   a. To enable Kerberos authentication for your Autonomous Database, you must obtain the Kerberos configuration files: `krb.conf` and the service key table file `v5srvtab`.

      To generate these files when you configure Kerberos authentication with CMU-AD you need the server host name. You can obtain the value of the server host from the attribute `PUBLIC_DOMAIN_NAME` in the `CLOUD_IDENTITY` column of `V$PDBS`. This value is different from the Fully Qualified Domain Name (FQDN) for a database on a private endpoint.

      Use the following command to obtain the server host name:

      ```
      SELECT guid ||'/'|| json_value(cloud_identity, '$.PUBLIC_DOMAIN_NAME')
          "KSERVICE/KINSTANCE" FROM v$pdbs;
      ```

      You can use a command such as the following to generate the service key table file:

      ```
      ktpass -princ ORACLE/
      DATABASE_SERVER_HOST_NAME.DATABASE_SERVER_HOST_DOMAIN@ACTIVE_DIRECTORY_D
      EFAULT_DOMAIN
                      -pass ACTIVE_DIRECTORY_PASSWORD
                      -mapuser DATABASE_SERVER_HOST_NAME
                      -crypto ALL
                      -ptype KRB5_NT_PRINCIPAL
                      -out database.keytab
      ```

      For example:

      ```
      ktpass -princ ORACLE/user.example.com@example.com
                      -pass password -mapuser dbexamplekrb
      ```

```
                    -crypto ALL -ptype KRB5_NT_PRINCIPAL -out
database.keytab
```

For more information on these files and steps required to obtain them, see Configuring Kerberos Authentication.

b. Copy the Kerberos configuration files `krb.conf` and `v5srvtab` to a bucket in your Object Store.

This step differs depending on the Object Store you use.

If you are using Oracle Cloud Infrastructure Object Store, see Putting Data into Object Storage for details on uploading files.

c. Run `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION` to enable Kerberos external authentication.

For example:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
       type     => 'KERBEROS',
       params   => JSON_OBJECT('location_uri' value 'https://
objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o',
                               'credential_name' value
'my_credential_name')
   );
END;
/
```

> **Note:**
>
> Oracle recommends that you store the Kerberos configuration files in a private bucket in your Object Store.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

The `credential_name` you use in this step is the credentials for the Object Store.

See Enable Kerberos Authentication on Autonomous Database for more information.

d. Confirm that Kerberos is configured and enabled.

```
SELECT property_value FROM database_properties
      WHERE property_name='KERBEROS_DIRECTORY';
```

2. Enable and configure CMU-AD on Autonomous Database.

a. Upload the CMU configuration files, including the database wallet file, `cwallet.sso` and the CMU configuration file, `dsi.ora` to your Object Store.

You upload the `cwallet.sso` so that the CMU-AD configuration has the credentials for the Autonomous Database instance to connect to the Active Directory service account.

The `dsi.ora` configuration file contains the information to find the Active Directory servers.

This step differs depending on the Object Store you use.

If you are using Oracle Cloud Infrastructure Object Store, see Putting Data into Object Storage for details on uploading files.

b. Run the `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION` procedure and pass in a location URI with the `params` JSON argument. You must place the configuration files `cwallet.sso` and `dsi.ora` in the Object Storage location specified in the `location_uri` parameter.

For example:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
       type     => 'CMU',
       params   => JSON_OBJECT('location_uri' value 'https://
objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o',
                                'credential_name' value
'my_credential_name')
   );
END;
/
```

Oracle recommends that you store the CMU configuration files in a private bucket in your Object Store.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

The `credential_name` you use in this step is the credentials to access the Object Store.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

If the `location_uri` is a pre-authenticated URL or a pre-signed URL, then supplying a `credential_name` is not required.

See Prerequisites to Configure CMU with Microsoft Active Directory on Autonomous Database for more information.

c. Confirm that CMU-AD is configured and enabled.

```
SELECT property_value FROM database_properties
      WHERE property_name='CMU_WALLET';
```

3. After completing Step 1 and Step 2, verify that the configuration of Kerberos authentication with CMU-AD is complete.

   a. Login to the Autonomous Database instance as an Active Directory user so that `SYS_CONTEXT` information populated in your `USERENV`.

b. Query the `SYS_CONTEXT USERENV`.

```
SELECT SYS_CONTEXT('USERENV','AUTHENTICATION_METHOD') FROM DUAL;

SYS_CONTEXT('USERENV','AUTHENTICATION_METHOD')

--------------------------------------------------------------------------------
--------
KERBEROS_GLOBAL
```

When you have Kerberos authentication configured and enabled without CMU-AD, this query returns: `KERBEROS`. See Configure Kerberos Authentication with Autonomous Database for more information.

When you have CMU-AD authentication configured without Kerberos, this query returns: `PASSWORD_GLOBAL`. See Prerequisites to Configure CMU with Microsoft Active Directory on Autonomous Database for more information.

Notes for using Kerberos authentication with CMU-AD:

- You do not need to add the password filter when using Kerberos authentication with CMU-AD. See Prerequisites to Configure CMU with Microsoft Active Directory on Autonomous Database for more information.
- Adding or removing Active Directory users is supported, in the same manner as with CMU with Active Directory when you are using password authentication. See Add Microsoft Active Directory Users on Autonomous Database for more information.
- The existing restrictions about authenticating against the Autonomous Database Built-in Tools with CMU with Active Directory password also apply to CMU with Active Directory with Kerberos authentication. See Tools Restrictions with Active Directory on Autonomous Database for more information.
- Use `DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION` to disable CMU-AD with Kerberos authentication. See DISABLE_EXTERNAL_AUTHENTICATION Procedure for more information.
- When the CMU-AD servers are on a private endpoint, to use CMU-AD with Kerberos authentication the server host name used for generating the key tab must be set to the value of attribute `PUBLIC_DOMAIN_NAME` in the `CLOUD_IDENTITY` column of `V$PDBS`. This value is different from the FQDN for a private endpoint database.

Add Microsoft Active Directory Roles on Autonomous Database

To add Active Directory roles, map the database global roles to Active Directory groups with `CREATE ROLE` or `ALTER ROLE` statements (and include the `IDENTIFIED GLOBALLY AS` clause).

To add global roles for Active Directory groups on Autonomous Database:

1. Log in as the ADMIN user to the database that is configured to use Active Directory (the ADMIN user has the `CREATE ROLE` and `ALTER ROLE` system privileges that you need for these steps).
2. Set the database authorization for Autonomous Database roles with `CREATE ROLE` or `ALTER ROLE` statement. Include the `IDENTIFIED GLOBALLY AS` clause and specify the DN of an Active Directory group.

Use the following syntax to map a directory user group to a database global role:

```
CREATE ROLE global_role IDENTIFIED GLOBALLY AS
     'DN_of_an_AD_GROUP_of_WHICH_the_AD_USER_IS_a_MEMBER';
```

For example:

```
CREATE ROLE widget_sales_role IDENTIFIED GLOBALLY AS
     'CN=widget_sales_group,OU=sales,DC=production,DC=example,DC=com';
```

In this example all members of the `widget_sales_group` are authorized with the database role `widget_sales_role` when they log in to the database.

3. Use `GRANT` statements to grant the required privileges or other roles to the global role.

   For example:

   ```
   GRANT CREATE SESSION TO WIDGET_SALES_ROLE;
   GRANT DWROLE TO WIDGET_SALES_ROLE;
   ```

   `DWROLE` is a predefined role that has common privileges defined. See Manage User Privileges on Autonomous Database - Connecting with a Client Tool for information on setting common privileges for Autonomous Database users.

4. If you want to make an existing database role to be associated with an Active Directory group, then use `ALTER ROLE` statement to alter the existing database role to map the role to an Active Directory group.

   Use the following syntax to alter an existing database role to map it to an Active Directory group:

   ```
   ALTER ROLE existing_database_role
      IDENTIFIED GLOBALLY AS
   'DN_of_an_AD_GROUP_of_WHICH_the_AD_USER_IS_a_MEMBER';
   ```

5. If you want to create additional global role mappings for other Active Directory groups, follow these steps for each Active Directory group.

See Configuring Authorization for Centrally Managed Users for more information on configuring roles with Microsoft Active Directory.

Add Microsoft Active Directory Users on Autonomous Database

To add Active Directory users to access a database, map database global users to Active Directory groups or users with `CREATE USER` or `ALTER USER` statements (with `IDENTIFIED GLOBALLY AS` clause).

The integration of Autonomous Database with Active Directory works by mapping Microsoft Active Directory users and groups directly to Oracle database global users and global roles.

To add global users for Active Directory groups or users on Autonomous Database:

1. Log in as the ADMIN user to the database that is configured to use Active Directory (the ADMIN user has the required `CREATE USER` and `ALTER USER` system privileges that you need for these steps).

2. Set database authorization for Autonomous Database users with `CREATE USER` or `ALTER USER` statements and include the `IDENTIFIED GLOBALLY AS` clause, specifying the DN of an Active Directory user or group.

   Use the following syntax to map a directory user to a database global user:

   ```
   CREATE USER global_user IDENTIFIED GLOBALLY AS 'DN_of_an_AD_USER';
   ```

   Use the following syntax to map a directory group to a database global user:

   ```
   CREATE USER global_user IDENTIFIED GLOBALLY AS
       'DN_of_an_AD_GROUP_of_WHICH_the_AD_USER_IS_a_MEMBER';
   ```

   For example, to map a directory group named `widget_sales_group` in the `sales` organization unit of the `production.example.com` domain to a shared database global user named `WIDGET_SALES`:

   ```
   CREATE USER widget_sales IDENTIFIED GLOBALLY AS
        'CN=widget_sales_group,OU=sales,DC=production,DC=example,DC=com';
   ```

   This creates a shared global user mapping. The mapping, with global user `widget_sales`, is effective for all users in the Active Directory group. Thus, anyone in the `widget_sales_group` can log in to the database using their Active Directory credentials (through the shared mapping of the `widget_sales` global user).

3. If you want Active Directory users to use an existing database user, own its schema, and own its existing data, then use `ALTER USER` to alter an existing database user to map the user to an Active Directory group or user.

   - Use the following syntax to alter an existing database user to map it to an Active Directory user:

     ```
     ALTER USER existing_database_user IDENTIFIED GLOBALLY AS
     'DN_of_an_AD_USER';
     ```

   - Use the following syntax to alter an existing database user to map it to an Active Directory group:

     ```
     ALTER USER existing_database_user
          IDENTIFIED GLOBALLY AS
     'DN_of_an_AD_GROUP_of_WHICH_the_AD_USER_IS_a_MEMBER';
     ```

4. If you want to create additional global user mappings for other Active Directory groups or users, follow these steps for each Active Directory group or user.

See Configuring Authorization for Centrally Managed Users for more information on configuring users with Microsoft Active Directory.

Tools Restrictions with Active Directory on Autonomous Database

Notes for using Autonomous Database tools with Active Directory:

- Oracle APEX is not supported for Active Directory users with Autonomous Database. See Create Oracle APEX Workspaces in Autonomous Database for information on using regular database users with Autonomous Database.

- Database Actions is not supported for Active Directory users with Autonomous Database. See Provide Database Actions Access to Database Users for information on using regular database users with Autonomous Database.
- Oracle Machine Learning Notebooks are not supported for Active Directory users with Autonomous Database. See Add Existing Database User Account to Oracle Machine Learning Components for information on using regular database users with Autonomous Database.

Connect to Autonomous Database with Active Directory User Credentials

After the ADMIN user completes the CMU Active Directory configuration steps and creates global roles and global users, users log in to the database using their Active Directory username and password.

> **Note:**
>
> Do not log in using a Global User name. Global User names do not have a password and connecting with a Global User name will not be successful. You must have a global user mapping in your Autonomous Database in order to log in to the database. You cannot log in to the database with only global role mappings.

- To log in to the database using an Active Directory username and password, connect as follows:

  ```
  CONNECT "AD_DOMAIN\AD_USERNAME"/
  AD_USER_PASSWORD@TNS_ALIAS_OF_THE_AUTONOMOUS_DATABASE;
  ```

  For example:

  ```
  CONNECT "production\pfitch"/password@adbname_medium;
  ```

  You need to include double quotes when the Active Directory domain is included along with the username, as with this example: `"production\pfitch"`.

  In this example, the Active Directory username is `pfitch` in domain `production`. The Active Directory user is a member of `widget_sales_group` group which is identified by its DN `'CN=widget_sales_group,OU=sales,DC=production,DC=example,DC=com'`.

After configuring CMU with Active Directory on Autonomous Database and setting up Active Directory authorization, with global roles and global users, you can connect to your database using any of the connection methods described in Connect to Autonomous Database. When you connect, if you want to use an Active Directory user then use Active Directory user credentials. For example, provide a username in this form, "*AD_DOMAIN\AD_USERNAME*" (double quotes must be included), and use your *AD_USER_PASSWORD* for the password.

If your Autonomous Database instance is in Restricted Mode, this mode only allows users with the `RESTRICTED SESSION` privilege to connect to the database. The ADMIN user has this privilege. You can use restricted access mode to perform administrative tasks such as indexing, data loads, or other planned activities. See Change Autonomous Database Operation Mode to Read/Write Read-Only or Restricted for more information.

Verify Active Directory User Connection Information with Autonomous Database

When users log in to the database using their Active Directory username and password, you can verify and audit the user activity.

For example, when the user `pfitch` logs in:

```
CONNECT "production\pfitch"/password@exampleadb_medium;
```

The Active Directory user's log on username (samAccountName) is `pfitch` and `widget_sales_group` is the Active Directory Group name, and `widget_sales` is the database global user.

After `pfitch` logs in to the database, the command `SHOW USER` shows the global user name:

```
SHOW USER;

USER is "WIDGET_SALES"
```

The following command shows the DN (Distinguished Name) of the Active Directory user:

```
SELECT SYS_CONTEXT('USERENV', 'ENTERPRISE_IDENTITY') FROM DUAL;
```

For example you can verify this centrally managed user's enterprise identity:

```
SQL> SELECT SYS_CONTEXT('USERENV', 'ENTERPRISE_IDENTITY') FROM DUAL;

SYS_CONTEXT('USERENV','ENTERPRISE_IDENTITY')
--------------------------------------------------------------------------------
cn=Peter Fitch,ou=sales,dc=production,dc=examplecorp,dc=com
```

The following command shows the "`AD_DOMAIN\AD_USERNAME`":

```
SELECT SYS_CONTEXT('USERENV', 'AUTHENTICATED_IDENTITY') FROM DUAL;
```

For example, the Active Directory authenticated user identity is captured and audited when the user logs on to the database:

```
SQL> SELECT SYS_CONTEXT('USERENV', 'AUTHENTICATED_IDENTITY') FROM DUAL;

SYS_CONTEXT('USERENV','AUTHENTICATED_IDENTITY')
--------------------------------------------------------------------------------
production\pfitch
```

See Verifying the Centrally Managed User Logon Information for more information.

Remove Active Directory Users and Roles on Autonomous Database

To remove Active Directory users and roles from Autonomous Databases, use standard database commands. This does not remove the related Active Directory users or groups that were mapped from the dropped database users or roles.

To remove users or roles from Autonomous Database:

1. Log in to the database that is configured to use Active Directory as a user who has been granted the `DROP USER` or `DROP ROLE` system privilege.
2. Drop the global users or the global roles that are mapped to Active Directory groups or users with `DROP USER` or `DROP ROLE` statement.

   See Remove Users on Autonomous Database for more information.

Disable Active Directory Access on Autonomous Database

Describes the steps to remove the CMU configuration from your Autonomous Database (and disable the LDAP access from your Autonomous Database to Active Directory).

After you configure your Autonomous Database instance to access CMU Active Directory, you can disable the access as follows:

1. Connect to the Autonomous Database as the ADMIN user.
2. Use the `DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION` to disable CMU authentication.

   > **Note:**
   >
   > To run this procedure you must be logged in as ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

   For example:

```
BEGIN
   DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
END;
/
```

   This disables CMU authentication on your Autonomous Database instance.

See DISABLE_EXTERNAL_AUTHENTICATION Procedure for more information.

Configure Kerberos Authentication with Autonomous Database

Describes how to configure Kerberos to authenticate Oracle Autonomous Database users.

- About Kerberos Authentication
  You can configure Oracle Autonomous Database to use Kerberos network authentication protocol to authenticate database users. Kerberos is a strong network authentication protocol. It uses secret-key cryptography to enable strong authentication by providing user-to-server authentication.

- Components of the Kerberos Authentication System
  Provides an overview of the Kerberos authentication system.
- Enable Kerberos Authentication on Autonomous Database
  Shows the steps to enable Kerberos authentication on your Autonomous Database instance.
- Disable Kerberos Authentication on Autonomous Database
  Shows the steps to disable Kerberos authentication for your Autonomous Database instance.
- Notes for Kerberos Authentication on Autonomous Database
  Provides notes on using Kerberos Authentication for Autonomous Database.

About Kerberos Authentication

You can configure Oracle Autonomous Database to use Kerberos network authentication protocol to authenticate database users. Kerberos is a strong network authentication protocol. It uses secret-key cryptography to enable strong authentication by providing user-to-server authentication.

- Oracle Autonomous Database support for Kerberos provides the benefits of single sign-on and centralized authentication of Oracle users. Kerberos is a trusted third-party authentication system that relies on shared secrets. It presumes that the third party is secure, and provides single sign-on capabilities, centralized password storage, database link authentication, and enhanced PC security. It does this through a Kerberos authentication server.
- The Kerberos system revolves around the concept of a ticket. A ticket is a set of electronic information that identifies a user or a service. A ticket identifies you and your network access privileges.
- In Kerberos-based authentication, you transparently send a request for a ticket to a Key Distribution Center (KDC). The Key Distribution Center authenticates you and grants you a ticket to access the database.

Components of the Kerberos Authentication System

Provides an overview of the Kerberos authentication system.

- A realm establishes an authentication administrative domain. Each realm has its own Kerberos database which contains the users and services for that particular administrative domain.
- Tickets are issued by the Key Distribution Center (KDC). Clients present tickets to the Database Server to demonstrate the authenticity of their identity. Each ticket has expiration and a renewal time.
- Keytabs stores long-term keys for one or more principals. A keytab file is generated by invoking the tool `kadmin.local` (for MIT Key Distribution Center) or `ktpass` (for Active Directory Key Distribution Center).
- Principals are the entries in the Key Distribution Center database. Each user, host or service is given a principal. A principal is a unique identity to which the Key Distribution Center can assign tickets.
- Kerberos support in Autonomous Database uses these values for various components that make up a service principal's name:

Component of Service Principal	Value in Autonomous Database
kinstance	You can obtain this value from the attribute PUBLIC_DOMAIN_NAME in the CLOUD_IDENTITY column of V$PDBS. This value is different from the Fully Qualified Domain Name (FQDN) for a database on a private endpoint. Use the following query to obtain the kinstance: SELECT json_value(cloud_identity, '$.PUBLIC_DOMAIN_NAME') "KINSTANCE" FROM v$pdbs;
kservice	On Autonomous Database you have two options for the kservice value: • Use the database GUID: If you do not provide a Kerberos service name with DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION, the default service name used is the GUID of the Autonomous Database instance. In this case, when you create the keytab file, use the GUID as the value for the service name. Because in the default case the Keytab file uses a service name that is a GUID, which is instance specific, when you use the default service name you must generate different Keytab files for each Autonomous Database instance. Use the following command to obtain the GUID (case is significant): SELECT GUID FROM v$pdbs; • Use a custom name: Set the service name when you want to use the same Keytab files on multiple Autonomous Database instances. When you use a custom name you do not need to create and upload different Keytab files for each Autonomous Database instance. When you use a custom name you must specify the params kerberos_service_name parameter with DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION. Specifying this parameter is not required when you use the GUID for the service name. After Kerberos is enabled on your Autonomous Database instance, use the following query to view the Kerberos service name: SELECT SYS_CONTEXT('USERENV', 'KERBEROS_SERVICE_NAME') FROM DUAL;
REALM	Any realm supported by your KDC. REALM must always be in uppercase.

To enable Kerberos authentication for your Autonomous Database, you must keep your Kerberos configuration files (krb.conf) and service key table file (v5srvtab) ready. For more information on these files and steps to obtain them, please see Configuring Kerberos Authentication.

Enable Kerberos Authentication on Autonomous Database

Shows the steps to enable Kerberos authentication on your Autonomous Database instance.

To run DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION you must be logged in as ADMIN user or have the EXECUTE privilege on DBMS_CLOUD_ADMIN.

To use DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION to enable Kerberos authentication:

To enable Kerberos authentication for your Autonomous Database, you must obtain the Kerberos configuration files: krb.conf and the service key table file v5srvtab. For more

information on these files and steps required to obtain them, see Configuring Kerberos Authentication .

1. Copy the Kerberos configuration files `krb.conf` and `v5srvtab` to a bucket in your Object Store.

   If you are using Oracle Cloud Infrastructure Object Store, see Putting Data into Object Storage for details on uploading files.

2. Run `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION` procedure and pass in a location URI with the `params` JSON argument. You must place the configuration files `krb.conf` and `v5srvtab` in the Object Storage location specified in the `location_uri` parameter.

   For example:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
       type     => 'KERBEROS',
       params   => JSON_OBJECT(
               'location_uri' value 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
               'credential_name' value 'my_credential_name'));
END;
/
```

> **Note:**
>
> Oracle recommends that you store the Kerberos configuration files in a private bucket in your Object Store.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

The `credential_name` you use in this step is the credentials for the Object Store.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

If the `location_uri` is a pre-authenticated URL then supplying a `credential_name` is not required.

This creates a directory object named `KERBEROS_DIR` in your database and uses the credential to download the Kerberos configuration files from the Object Store location to the directory object.

You can specify the `params kerberos_service_name` parameter to specify a Kerberos service name. For example:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
       type     => 'KERBEROS',
       params   => JSON_OBJECT(
               'location_uri' value 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
```

```
                'credential_name' value 'my_credential_name'
                'kerberos_service_name' value 'oracle' ));
END;
/
```

3. After you enable Kerberos authentication, remove the configuration `krb.conf` and `v5srvtab` from Object Store. You can use local Object Store methods to remove these files or use `DBMS_CLOUD.DELETE_OBJECT` to delete the files from Object Store.

See Navigate to Oracle Cloud Infrastructure Object Storage and Create Bucket for more information on Object Storage.

See ENABLE_EXTERNAL_AUTHENTICATION Procedure for more information.

Disable Kerberos Authentication on Autonomous Database

Shows the steps to disable Kerberos authentication for your Autonomous Database instance.

- Run `DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION` procedure to disable Kerberos authentication. To run the procedure, you must be logged in as ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

  ```
  BEGIN
     DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
  END;
  /
  ```

  This disables the Kerberos authentication (or any external authentication scheme specified) for Oracle Autonomous Database.

  See DISABLE_EXTERNAL_AUTHENTICATION Procedure for more information.

Notes for Kerberos Authentication on Autonomous Database

Provides notes on using Kerberos Authentication for Autonomous Database.

- If you enable Kerberos authentication for your Autonomous Database, you can still use password-based database authentication for your database.
- Kerberos authentication is not supported for these tools:
  - Oracle Database API for MongoDB
  - Oracle REST Data Services
  - Oracle Machine Learning
  - APEX
  - Oracle Graph Studio
  - Oracle Database Actions
- You can enable Kerberos authentication to authenticate the ADMIN user. You can use the Reset Password functionality on the Oracle Cloud Infrastructure Console to reset the ADMIN user's password and regain access if a corrupted keytab file causes ADMIN user's authentication to fail.
- The default value for the maximum **clock skew** in Autonomous Database is 300 seconds (5 minutes). You cannot change the default **clock skew** value.

Manage Credentials or Configure Policies and Roles to Access Resources

Describes the methods in which Autonomous Database connects to other resources, either with credentials, or by configuring policies and roles for access.

- Manage Credentials
  You can create credentials, list credentials, or delete credentials in your Autonomous Database.
- Use Vault Secret Credentials
- Configure Policies and Roles to Access Resources
  You may use the following methods to access cloud resources securely without storing user credentials: Oracle Cloud Infrastructure Resource Principals, AWS Amazon Resource Names (ARN)s, Azure service principal, or Google service accounts.

Manage Credentials

You can create credentials, list credentials, or delete credentials in your Autonomous Database.

- Create Credentials to Access Cloud Services
  To access services in the Cloud, such as Cloud Object Store, you first need to create credentials in your Autonomous Database.
- List Credentials
  `DBMS_CLOUD` provides the ability to store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`. You can list credentials from the view `ALL_CREDENTIALS`.
- Delete Credentials
  `DBMS_CLOUD` provides the ability to store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`. You can remove credentials with `DBMS_CLOUD.DROP_CREDENTIAL`.

Create Credentials to Access Cloud Services

To access services in the Cloud, such as Cloud Object Store, you first need to create credentials in your Autonomous Database.

1. Create and store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`. For example:

```
SET DEFINE OFF
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adb_user@example.com',
    password => 'password'
  );
END;
/
```

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for all data loads.

For detailed information about the parameters, see CREATE_CREDENTIAL Procedure.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

> **Note:**
>
> Some tools like SQL*Plus and SQL Developer use the ampersand character (`&`) as a special character. If you have the ampersand character in your password use the `SET DEFINE OFF` command in those tools as shown in the example to disable the special character and get the credential created properly.

2. With the credential you created in Step 1, you can access Object Store or other cloud resources from Autonomous Database using a procedure such as `DBMS_CLOUD.COPY_DATA`, `DBMS_CLOUD.EXPORT_DATA`, `DBMS_CLOUD_PIPELINE` if you are using a Data Pipeline, or other procedures that require `DBMS_CLOUD` credentials.

List Credentials

`DBMS_CLOUD` provides the ability to store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`. You can list credentials from the view `ALL_CREDENTIALS`.

For example, to list credentials, run the following command:

```
SELECT credential_name, username, comments FROM all_credentials;

CREDENTIAL_NAME                                            USERNAME
---------------------------------------------------------  --------------------
COMMENTS
---------------------------------------------------------  --------------------
ADB_TOKEN                                                  user_name@example.com
{"comments":"Created via DBMS_CLOUD.create_credential"}
DEF_CRED_NAME                                              user_name@example.com
{"comments":"Created via DBMS_CLOUD.create_credential"}
```

See ALL_CREDENTIALS for more information.

Delete Credentials

`DBMS_CLOUD` provides the ability to store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`. You can remove credentials with `DBMS_CLOUD.DROP_CREDENTIAL`.

For example, to remove the credential named `DEF_CRED_NAME`, run the following command:

```
BEGIN
   DBMS_CLOUD.DROP_CREDENTIAL('DEF_CRED_NAME');
END;
```

For more information about the `DBMS_CLOUD` procedures and parameters, see DBMS_CLOUD Subprograms and REST APIs.

Use Vault Secret Credentials

Describes using vault secret credentials, where the credentials secret (password) is stored as a secret in a vault. You can then use vault secret credentials to access cloud resources or to access other databases (use anywhere that username/password type credentials are required).
You can create vault secret credentials with secrets stored in any of the supported vaults:

- Oracle Cloud Infrastructure Vault
- Azure Key Vault
- AWS Secrets Manager
- GCP Secret Manager

For example, some possible uses cases for vault secret credentials:

- You can avoid duplicating secrets (passwords) when you access cloud resources from an Autonomous Database instance. In this case, you store secrets in a vault and Autonomous Database accesses the vault. This allows you to rotate secrets without updating the credentials you create to access cloud resources.
- You can use vault secret credentials with database links. In this case, you can create routines that access another database and you don't need to expose passwords in your code.
- You can use vault secret credentials with DBMS_DATAPUMP APIs.

> **Note:**
>
> Operations that use Oracle Data Pump support vault secret credentials (for example `impdp` and `expdp`). Oracle Data Pump support with vault secret credentials is limited to Oracle Cloud Infrastructure Swift URIs and Oracle Cloud Infrastructure Native URIs. See DBMS_CLOUD URI Formats for more information.

Topics

- Use Vault Secret Credentials with Oracle Cloud Infrastructure Vault
  Describes using vault secret credentials, where the secret (password) is stored as a secret in Oracle Cloud Infrastructure Vault.
- Use Vault Secret Credential with Azure Key Vault
  Describes using vault secret credentials, where the credential secret (password) is stored in Azure Key Vault.
- Use Vault Secret Credential with AWS Secrets Manager
  Describes using vault secret credentials, where the credential secret (password) is stored in AWS Secrets Manager.
- Use Vault Secret Credential with GCP Secret Manager
  Describes using vault secret credentials, where the credentials secret (password) are stored as a secret in GCP Secret Manager.
- Refresh Vault Secret Credentials
- Update Vault Secret Credentials

Use Vault Secret Credentials with Oracle Cloud Infrastructure Vault

Describes using vault secret credentials, where the secret (password) is stored as a secret in Oracle Cloud Infrastructure Vault.

You can use vault secret credentials to access cloud resources, to access other databases with database links, or use anywhere that username/password type credentials are required.

- Prerequisites to Create Vault Secret Credentials with Oracle Cloud Infrastructure Vault
- Create Vault Secret Credentials with Oracle Cloud Infrastructure Vault

Prerequisites to Create Vault Secret Credentials with Oracle Cloud Infrastructure Vault

Describes the required prerequisite steps to use vault secret credentials with Oracle Cloud Infrastructure Vault secrets.

To create vault secret credentials where the secret is stored in Oracle Cloud Infrastructure Vault, first perform the required prerequisites.

1. Create a vault and create a secret in the vault with Oracle Cloud Infrastructure Vault.

   For more information, see the instructions for creating a vault and a secret, Managing Vaults and Overview of Key Management.

2. Set up a dynamic group to provide access to the secret in the Oracle Cloud Infrastructure Vault.

   Create a dynamic group for the Autonomous Database instance where you want to create a vault secret credential:

   a. In the Oracle Cloud Infrastructure console click **Identity & Security**.
   b. Under **Identity** click **Domains** and select an identity domain (or create a new identity domain).
   c. Under Identity domain, click **Dynamic groups**.
   d. Click **Create dynamic group** and enter a **Name**, a **Description**, and a rule.
      - **Create Dynamic Group for an existing database:**

You can specify that an Autonomous Database instance is part of the dynamic group. The dynamic group in the following example includes only the Autonomous Database whose OCID is specified in the `resource.id` parameter:

```
resource.id = 'your_Autonomous_Database_instance_OCID'
```

- **Create a Dynamic Group for a database that has not been provisioned yet:**

  When you are creating the dynamic group before you provision or clone an Autonomous Database instance, the OCID for the new database is not yet available. For this case, create a dynamic group that specifies the resources in a given compartment:

```
resource.compartment.id = 'your_Compartment_OCID'
```

e. Click **Create**.

3. Write policy statements for the dynamic group to enable access to Oracle Cloud Infrastructure resources (secrets).

   a. In the Oracle Cloud Infrastructure Console click **Identity and Security** and click **Policies**.

   b. To write policies for the dynamic group you created in the previous step, click **Create Policy**, and enter a **Name** and a **Description**.

   c. Use the **show manual editor** option of **Policy Builder** to create a policy.

   For example, to allow access to the dynamic group to read a specific secret in a compartment:

```
Allow dynamic-group dynamic_group_name to read secret-bundles in
compartment compartment_name
   where target.secret.id='secret_OCID'
```

   For example, to allow access to the dynamic group to read all secrets in a compartment:

```
Allow dynamic-group dynamic_group_name to read secret-bundles in
compartment compartment_name
```

   See Details for the Vault Service for more information.

   d. Select the group or dynamic group and select the location.

   e. Click **Create**.

Create Vault Secret Credentials with Oracle Cloud Infrastructure Vault

Describes the steps to use an Oracle Cloud Infrastructure Vault secret with credentials.

This allows you to store a secret in Oracle Cloud Infrastructure Vault and use the secret with the credential you create to access cloud resources or to access other databases.

To create vault secret credentials where the secret is stored in Oracle Cloud Infrastructure Vault:

1. Enable resource principal authentication to provide access to a secret in the Oracle Cloud Infrastructure Vault.

See Enable Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

2. Create a dynamic group and define policies to allow your Autonomous Database to access secrets in an Oracle Cloud Infrastructure Vault.

   See Prerequisites to Create Vault Secret Credentials with Oracle Cloud Infrastructure Vault for more information.

3. Use `DBMS_CLOUD.CREATE_CREDENTIAL` to create a vault secret credential.

   For example:

```
BEGIN DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name   => 'OCI_SECRET_CRED',
    params            => JSON_OBJECT(
        'username'   value 'SCOTT',
        'secret_id'  value
'ocid1.vaultsecret.oc1.iad.example..aaaaaaaauq5ok5nq3bf2vwetkpqsoa' ));
END;
/
```

   Where:

   - `username`: is the username of the original credential. It can be the username of any type of username/password credential such as the username of an OCI Swift user, the username required to access a database with a database link, and so on.
   - `secret_id`: is the vault secret ID. For example, when you store the password *mysecret* in a secret in Oracle Cloud Infrastructure Vault, the `secret_id` value is the vault secret OCID.

   To create a vault secret credential you must have `EXECUTE` privilege on the `DBMS_CLOUD` package.

   See CREATE_CREDENTIAL Procedure for more information.

4. Use the credential to access a cloud resource.

   For example:

```
SELECT count(*) FROM DBMS_CLOUD.LIST_OBJECTS(
    'OCI_SECRET_CRED',
    'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/' );
```

> **Note:**
>
> Every 12 hours the secret (password) is refreshed from the content in the Oracle Cloud Infrastructure Vault. If you change the secret value in the Oracle Cloud Infrastructure Vault, it can take up to 12 hours for the Autonomous Database instance to pick up the latest secret value.

Run `DBMS_CLOUD.REFRESH_VAULT_CREDENTIAL` to immediately refresh a vault secret credential. This procedure gets the latest version of the vault secret from Oracle Cloud Infrastructure Vault. See REFRESH_VAULT_CREDENTIAL Procedure for more information.

Use Vault Secret Credential with Azure Key Vault

Describes using vault secret credentials, where the credential secret (password) is stored in Azure Key Vault.

You can use vault secret credentials to access cloud resources, to access other databases with database links, or use anywhere that username/password type credentials are required.

- Prerequisites to Create Vault Secret Credential with Azure Key Vault
  Describes the required prerequisites to use vault secret credentials with Azure Key Vault.
- Create Vault Secret Credential with Azure Key Vault
  Describes the steps to use an Azure Key Vault with vault secret credentials.

Prerequisites to Create Vault Secret Credential with Azure Key Vault

Describes the required prerequisites to use vault secret credentials with Azure Key Vault.

To create vault secret credentials where the secret is stored in Azure Key Vault, first perform the required prerequisites.

1. Enable Azure service principal authentication to provide access to the key (password) in the Azure Key Vault.

   See Enable Azure Service Principal for more information.
2. Create an Azure Key Vault and create a secret (password) in the vault.

   See About Azure Key Vault for more information.
3. Set up and enable Azure Service Principal to provide access to the secret in the Azure Key Vault.

   In the Azure portal you must grant read access for a service principal to access the secret.

   a. In the Azure portal, navigate to the "Key Vault" resource that contains the secret you created.
   b. Select "Access policies", then select **Create**.
   c. Under Permissions, select the **Get** permission in the **Secret permissions** section.
   d. Under Principal, enter the name of the service principal in the search field and select the appropriate result.
   e. Click **Next**.
   f. Under Review + create, review the access policy changes and click **Create** to save the access policy.
   g. Back on the Access policies page, verify that your access policy is listed.

   See Assign a Key Vault access policy for more information.

Create Vault Secret Credential with Azure Key Vault

Describes the steps to use an Azure Key Vault with vault secret credentials.

This allows you to store a secret in Azure Key Vault and use the secret with the credentials you create to access cloud resources or to access other databases.

To create vault secret credentials where the secret is stored in Azure Key Vault:

1. Create the Azure Key Vault, the secret, and the access policies to allow your Autonomous Database to access secrets in an Azure Key Vault.

   See Prerequisites to Create Vault Secret Credential with Azure Key Vault for more information.

2. Use `DBMS_CLOUD.CREATE_CREDENTIAL` to create a vault secret credential.

   For example:

```
BEGIN DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name  => 'AZURE_SECRET_CRED',
    params  => JSON_OBJECT(
      'username'          value 'azure_user',
      'secret_id'         value 'sales-secret',
      'azure_vault_name'  value 'azure_keyvault_name' ));
END;
/
```

   Where:

   - `username`: is the username of the original credential. It can be the username of any type of username/password credential.
   - `secret_id`: is the secret name.
   - `azure_vault_name`: is the name of the vault where the secret is located.

   To create a vault secret credential you must have `EXECUTE` privilege on the `DBMS_CLOUD` package.

   See CREATE_CREDENTIAL Procedure for more information.

3. Use the credential to access a cloud resource.

   For example:

```
SELECT count(*) FROM DBMS_CLOUD.LIST_OBJECTS(
     'AZURE_SECRET_CRED',
     'https://adb_user.blob.core.windows.net/adb/' );
```

> **Note:**
>
> Every 12 hours the secret (password) is refreshed from the content in the Azure Key Vault. If you change the secret value in the Azure Key Vault, it can take up to 12 hours for the Autonomous Database instance to pick up the latest secret value.

Run `DBMS_CLOUD.REFRESH_VAULT_CREDENTIAL` to immediately refresh a vault secret credential. This procedure gets the latest version of the vault secret from Azure Key Vault. See REFRESH_VAULT_CREDENTIAL Procedure for more information.

Use Vault Secret Credential with AWS Secrets Manager

Describes using vault secret credentials, where the credential secret (password) is stored in AWS Secrets Manager.

You can use vault secret credentials to access cloud resources, to access other databases with database links, or use anywhere that username/password type credentials are required.

- Prerequisites to Create Vault Secret Credential with AWS Secrets Manager
  Describes the required prerequisites to use vault secret credentials with AWS Secrets Manager.
- Create Vault Secret Credential with AWS Secrets Manager
  Describes the steps to use an AWS Secrets Manager secret with credentials.

Prerequisites to Create Vault Secret Credential with AWS Secrets Manager

Describes the required prerequisites to use vault secret credentials with AWS Secrets Manager.

To create vault secret credentials where the secret is stored in AWS Secrets Manager, first perform the required prerequisites.

1. Create a secret with AWS Secrets Manager and copy the AWS secret ARN.

   See AWS Secrets Manager for more information.

2. Perform AWS management prerequisites to use Amazon Resource Names (ARNs).

   See Perform AWS Management Prerequisites to Use Amazon Resource Names (ARNs) for more information.

3. Enable AWS principal authentication to provide access to the secret in AWS Secrets Manager.

   For example, on the Autonomous Database instance run:

   ```
   BEGIN

   DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
             provider => 'AWS',
             params =>
                JSON_OBJECT(
                   'aws_role_arn' value 'arn:aws:iam::123456:role/
   AWS_ROLE_ARN'));
   END;
   /
   ```

   See ENABLE_PRINCIPAL_AUTH Procedure and About Using Amazon Resource Names (ARNs) to Access AWS Resources for more information.

4. Set up AWS to provide permissions for Amazon Resource Names (ARNs) to access the secret in AWS Secrets Manager.

   On the AWS console, you must grant read access to the secret to the principal authentication credential.

   a. In the AWS console, navigate to the IAM and select **Roles** in Access Management.
   b. Select the role.
   c. In the Permission tab, click **Add permissions → create inline policy**.
   d. In the Service section, select **secret manager** as the service.
   e. In the Action section, select **Read** access level.

f. In the Resources section, click **Add ARN**, specify ARN for secret and click **Add** → click **Review the policy** → give a policy name → click **create policy**.

g. Back to the Permission tab, verify that the inline policy is attached.

See Use Amazon Resource Names (ARNs) to Access AWS Resources and Permissions policy examples for AWS Secrets Manager for more information.

Create Vault Secret Credential with AWS Secrets Manager

Describes the steps to use an AWS Secrets Manager secret with credentials.

This allows you to store a secret in AWS Secrets Manager and use the secret with the credentials you create to access cloud resources or to access other databases.

To create vault secret credentials where the secret is stored in AWS Secrets Manager:

1. Create a secret in AWS Secrets Manager and create an inline policy to allow your Autonomous Database to access secrets in AWS Secrets Manager.

   See Prerequisites to Create Vault Secret Credential with AWS Secrets Manager for more information.

2. Use `DBMS_CLOUD.CREATE_CREDENTIAL` to create a vault secret credential to access the AWS Secrets Manager secret.

   For example:

   ```
   BEGIN DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name      => 'AWS_SECRET_CRED',
       params               => JSON_OBJECT(
            'username'   value 'access_key',
            'secret_id'  value 'arn:aws:secretsmanager:region:account-
   ID:secret:secret_name' ));
   END;
   /
   ```

   Where:

   - `username`: is the username of the original credential. It can be the username of any type of username/password credential.
   - `secret_id`: is the vault secret AWS ARN.

   To create a vault secret credential you must have `EXECUTE` privilege on the `DBMS_CLOUD` package.

   See CREATE_CREDENTIAL Procedure for more information.

3. Use the credential to access a cloud resource.

   For example:

   ```
   SELECT count(*) FROM DBMS_CLOUD.LIST_OBJECTS(
        'AWS_SECRET_CRED',
        'https://s3-us-west-2.amazonaws.com/adb/' );
   ```

> **Note:**
>
> Every 12 hours the secret (password) is refreshed from the content in the AWS Secrets Manager. If you change the secret value in the AWS Secrets Manager, it can take up to 12 hours for the Autonomous Database instance to pick up the latest secret value.

Run `DBMS_CLOUD.REFRESH_VAULT_CREDENTIAL` to immediately refresh a vault secret credential. This procedure gets the latest version of the vault secret from AWS Secrets Manager. See REFRESH_VAULT_CREDENTIAL Procedure for more information.

Use Vault Secret Credential with GCP Secret Manager

Describes using vault secret credentials, where the credentials secret (password) are stored as a secret in GCP Secret Manager.

You can use vault secret credentials to access cloud resources, to access other databases with database links, or use anywhere that username/password type credentials are required.

- Prerequisites to Create Vault Secret Credential with GCP Secret Manager
  Describes the required prerequisites to use vault secret credentials with GCP Secret Manager.
- Create Vault Secret Credential with GCP Secret Manager
  Describes the steps to use an GCP Secret Manager secret to store secrets for use with the credentials you use to access cloud resources.

Prerequisites to Create Vault Secret Credential with GCP Secret Manager

Describes the required prerequisites to use vault secret credentials with GCP Secret Manager.

To create vault secret credentials where the secret is stored in GCP Secret Manager, first perform the required prerequisites.

1. Create a secret in GCP Secret Manager.

   See Secret Manager and Create and access a secret using Secret Manager for more information.

2. Enable Google Service Account authentication to provide access to GCP Secret Manager.

   On the Google Cloud console you must grant read access to the secret to the principal authentication credential.

   a. Go to the **Secret Manager** page in the Google Cloud console.

   b. On the **Secret Manager** page, click the checkbox next to the name of the secret.

   c. If it is not already open, click **Show Info Panel** to open the panel.

   d. In the info panel, click **Add Principal**.

   e. In the **New principals** text area, enter the service account name to add.

   f. In the **Select a role** dropdown, choose **Secret Manager** and then **Secret Manager Secret Accessor**.

   See Enable Google Service Account and Find the GCP Service Account Name and Manage access to secrets for more information.

Create Vault Secret Credential with GCP Secret Manager

Describes the steps to use an GCP Secret Manager secret to store secrets for use with the credentials you use to access cloud resources.

This allows you to store a secret in GCP Secret Manager and use the secret with the credentials you create to access cloud resources or to access other databases.

To create vault secret credentials where the secret is stored in GCP Secret Manager:

1. Create a secret manager secret accessor to allow your Autonomous Database principal to access secrets in GCP Secret Manager.

   See Prerequisites to Create Vault Secret Credential with GCP Secret Manager for more information.

2. Enable Google service account based authentication to provide access to the secret in the GCP Secret Manager.

   See Enable Google Service Account and Find the GCP Service Account Name for more information.

3. Use `DBMS_CLOUD.CREATE_CREDENTIAL` to create a vault secret credential to access the GCP Secret Manager secret.

   For example:

   ```
   BEGIN DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name      => 'GCP_SECRET_CRED',
       params               => JSON_OBJECT(
             'username'   value 'gcp_user1',
             'secret_id'  value 'my-secret',
             'gcp_project_id' value 'my-sample-project-191923' ));
   END;
   /
   ```

   Where:

   - `username`: is the username of the original credential. It can be the username of any type of username/password credential.
   - `secret_id`: is the secret name. When you store the password *mysecret* in the vault, use the secret name as the value of the `secret_id` parameter.
   - `gcp_project_id`: is the ID of the project where the secret is located.

   See CREATE_CREDENTIAL Procedure for more information.

4. Use the credential to access a cloud resource.

   For example:

   ```
   SELECT count(*) FROM DBMS_CLOUD.LIST_OBJECTS(
              'GCP_SECRET_CRED',
              'https://bucketname.storage.googleapis.com/' );
   ```

> **Note:**
>
> Every 12 hours the secret (password) is refreshed from the content in the GCP Secret Manager. If you change the secret value in the GCP Secret Manager, it can take up to 12 hours for the Autonomous Database instance to pick up the latest secret value.

Run `DBMS_CLOUD.REFRESH_VAULT_CREDENTIAL` to immediately refresh a vault secret credential. This procedure gets the latest version of the vault secret from GCP Secret Manager. See REFRESH_VAULT_CREDENTIAL Procedure for more information.

Refresh Vault Secret Credentials

Describes how vault secret credentials can be refreshed from the latest value in the vault.

To refresh vault secret credentials, you have two choices:

- Wait for automatic refresh: Every 12 hours the secret (password) is refreshed from the content in the vault. If you change the secret value in the vault, it can take up to 12 hours for the Autonomous Database instance to pick up the latest secret value.
- Run `DBMS_CLOUD.REFRESH_VAULT_CREDENTIAL` to immediately refresh a vault secret credential. This procedure gets the latest version of the vault secret from Oracle Cloud Infrastructure Vault.

  For example:

```
BEGIN
  DBMS_CLOUD.REFRESH_VAULT_CREDENTIAL(
     credential_name => 'VAULT_SECRET_CRED');
END;
/
```

  See REFRESH_VAULT_CREDENTIAL Procedure for more information.

Update Vault Secret Credentials

Describes the steps to update vault secret credential attributes.

To update vault secret credential attributes:

1. Use `DBMS_CLOUD.UPDATE_CREDENTIAL` to update a vault secret credential attribute.

   For example:

```
BEGIN DBMS_CLOUD.UPDATE_CREDENTIAL(
    credential_name   => 'VAULT_SECRET_CRED',
    attribute         => 'secret_id',
    value             => 'new_secret_id'
  );
END;
/
```

   To update a vault secret credential you must have `EXECUTE` privilege on the `DBMS_CLOUD` package.

See UPDATE_CREDENTIAL Procedure for more information.

2. Use the updated credential to access a cloud resource.

   For example:

```
SELECT count(*) FROM DBMS_CLOUD.LIST_OBJECTS(
    'VAULT_SECRET_CRED',
    'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/' );
```

Configure Policies and Roles to Access Resources

You may use the following methods to access cloud resources securely without storing user credentials: Oracle Cloud Infrastructure Resource Principals, AWS Amazon Resource Names (ARN)s, Azure service principal, or Google service accounts.

- Use Resource Principal to Access Oracle Cloud Infrastructure Resources
  You can use an Oracle Cloud Infrastructure resource principal with Autonomous Database. You or your tenancy administrator define the Oracle Cloud Infrastructure policies and a dynamic group that allows you to access Oracle Cloud Infrastructure resources with a resource principal. You do not need to create a credential object and Autonomous Database creates and secures the resource principal credentials you use to access the specified Oracle Cloud Infrastructure resources.
- Use Amazon Resource Names (ARNs) to Access AWS Resources
  You can use Amazon Resource Names (ARNs) to access AWS resources with Autonomous Database.
- Use Azure Service Principal to Access Azure Resources
  You can use an Azure service principal with Autonomous Database to access Azure resources without having to create and store your own credential objects in the database.
- Use Google Service Account to Access Google Cloud Platform Resources
  You can use a Google service account to access Google Cloud Platform (GCP) resources from an Autonomous Database instance.

Use Resource Principal to Access Oracle Cloud Infrastructure Resources

You can use an Oracle Cloud Infrastructure resource principal with Autonomous Database. You or your tenancy administrator define the Oracle Cloud Infrastructure policies and a dynamic group that allows you to access Oracle Cloud Infrastructure resources with a resource principal. You do not need to create a credential object and Autonomous Database creates and secures the resource principal credentials you use to access the specified Oracle Cloud Infrastructure resources.

- About Using Resource Principal to Access Oracle Cloud Infrastructure Resources
  You can use a resource principal to authenticate and access Oracle Cloud Infrastructure resources.
- Perform Prerequisites to Use Resource Principal with Autonomous Database
  Prior to making a call to an Oracle Cloud Infrastructure resource using a resource principal, an Oracle Cloud Infrastructure tenancy administrator must create Oracle Cloud Infrastructure policies, dynamic groups, and rules that define the resource principal privileges.
- Enable Resource Principal to Access Oracle Cloud Infrastructure Resources
  Perform the following steps to enable resource principal on Autonomous Database.

- Disable Resource Principal on Autonomous Database
  Shows the steps to disable resource principal for all Autonomous Database users or for a specified user.
- Use Resource Principal with DBMS_CLOUD
  When you specify resource principal credentials with `DBMS_CLOUD` calls, the database authenticates the Oracle Cloud Infrastructure requests for you and the database provides the credentials to access Oracle Cloud Infrastructure resources.

About Using Resource Principal to Access Oracle Cloud Infrastructure Resources

You can use a resource principal to authenticate and access Oracle Cloud Infrastructure resources.

A resource principal consists of a temporary session token and secure credentials that enable the database to authenticate itself to other Oracle Cloud Infrastructure services. Using a resource principal to access services, the token stored with the credentials on Autonomous Database is only valid for the resources to which the dynamic group has been granted access.

To use Resource Principal, you or your tenancy administrator define the Oracle Cloud Infrastructure policies and a dynamic group that allows you to access Oracle Cloud Infrastructure resources with a resource principal. You do not need to create a credential object and Autonomous Database creates and secures the resource principal credentials you use to access the specified Oracle Cloud Infrastructure resources.

For example, while using Autonomous Database you might want to use Oracle Cloud Infrastructure resources to do the following:

- Access data from an Object Storage bucket, perform some operation on the data, and then write the modified data back to the Object Storage bucket.
- Access your vaults, keys, or secrets.
- List work requests or list work request errors.

When you are working with the database, you authenticate and access the database as a database user. An Autonomous Database user does not have an Oracle Cloud Infrastructure Identity and Access Management (IAM) identity, so as an Autonomous Database user you cannot use your database credentials to access Oracle Cloud Infrastructure services. Without a resource principal you must obtain credentials to access Oracle Cloud Infrastructure resources and create a credential object to access a resource from Autonomous Database.

A resource principal enables resources to be authorized to perform actions on Oracle Cloud Infrastructure services. Each resource has its own identity, and the resource authenticates using the certificates that are added to it. These certificates are automatically created, assigned to resources, and rotated, avoiding the need for you to create and manage your own credentials to access the resource.

Autonomous Database lets you use a resource principal to authenticate to Oracle Cloud Infrastructure APIs using the following interfaces:

- `DBMS_CLOUD` procedures and functions that take a credential argument
- Oracle Cloud Infrastructure PL/SQL SDK APIs

When you authenticate using a resource principal, Autonomous Database provides a secure method to access Oracle Cloud Infrastructure resources.

There are several steps required to set up a resource principal on Autonomous Database:

- You must create an define Oracle Cloud Infrastructure Infrastructure Identity and Access Management (IAM) policies. See Perform Prerequisites to Use Resource Principal with Autonomous Database for more information.
- You must enable the resource principal for the ADMIN user, and optionally enable the resource principal for a database user. See Enable Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

When you authenticate using a resource principal, you do not need to create and manage credentials to access Oracle Cloud Infrastructure resources. Autonomous Database makes the resource principal available to you and secures the resource principal for you.

Perform Prerequisites to Use Resource Principal with Autonomous Database

Prior to making a call to an Oracle Cloud Infrastructure resource using a resource principal, an Oracle Cloud Infrastructure tenancy administrator must create Oracle Cloud Infrastructure policies, dynamic groups, and rules that define the resource principal privileges.

Perform the following steps before you use a resource principal with Autonomous Database:

1. Create an Oracle Cloud Infrastructure dynamic group.
   a. In the Oracle Cloud Infrastructure console click **Identity and Security** and click **Dynamic Groups**
   b. Click **Create Dynamic Group** and enter a **Name**, a **Description**, and a rule or use the **Rule Builder** to add a rule.
   c. Click **Create**.

   Resources that meet the rule criteria are members of the dynamic group. When you define a rule for a dynamic group, consider what resource is going to be given access to other resources.

   For example, consider the following examples:

   - Allow a specific Autonomous Database instance to access a resource.

     The Autonomous Database is specified in the `resource.id` parameter with an OCID:

     ```
     resource.id = '<your_Autonomous_Database_instance_OCID>'
     ```

   - Allow all Autonomous Databases in a compartment.

     The Autonomous Databases are specified in the `resource.type` parameter and the compartment is identified by a specified OCID in the `resource.compartment.id` parameter:

     ```
     ALL {resource.type = 'autonomousdatabase', resource.compartment.id =
     '<your_Compartment_OCID>'}
     ```

   - Allow all resources in the compartment

     The resource type identified by the OCID, specified in the `resource.compartment.id` parameter:

     ```
     ALL
             {resource.compartment.id='<your_Compartment_OCID>'}
     ```

   See Managing Dynamic Groups for more information on creating a dynamic group and creating rules to add resources to the group.

2. Write policy statements for the dynamic group to enable access to Oracle Cloud Infrastructure resources.
   a. In the Oracle Cloud Infrastructure console click **Identity and Security** and click **Policies**.
   b. To write policies for a dynamic group, click **Create Policy**, and enter a **Name** and a **Description**.
   c. Use the **Policy Builder** to create a policy.

      For example to create a policy to allow access to Oracle Cloud Infrastructure Object Store to manage buckets and objects in the Object Store in a tenancy:

      ```
      Allow dynamic-group Example5 to manage buckets in tenancy
      Allow dynamic-group Example5 to manage objects in tenancy
      ```

   d. Click **Create**.

      See Managing Policies for more information on policies.

> **Note:**
>
> The resource principal token is cached for two hours. Therefore, if you change the policy or the dynamic group, you have to wait for two hours to see the effect of your changes.

Enable Resource Principal to Access Oracle Cloud Infrastructure Resources

Perform the following steps to enable resource principal on Autonomous Database.

As a prerequisite, configure dynamic groups and policies. See Perform Prerequisites to Use Resource Principal with Autonomous Database for more information.

To enable a resource principal on Autonomous Database:

1. As the ADMIN user, enable resource principal for the Autonomous Database instance.

   For example:

   ```
   EXEC DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL();

   PL/SQL procedure successfully completed.
   ```

   See ENABLE_RESOURCE_PRINCIPAL Procedure for more information.

   This creates the credential `OCI$RESOURCE_PRINCIPAL`.

2. (Optional) This step is only required if you want to grant access to the resource principal credential to a database user other than the ADMIN user. As the ADMIN user, enable resource principal for a specified database user.

   For example:

   ```
   EXEC DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL(username => 'adb_user');

   PL/SQL procedure successfully completed.
   ```

This grants the user `adb_user` access to the credential `OCI$RESOURCE_PRINCIPAL`.

If you want the specified user to have privileges to enable resource principal for other users, set the `grant_option` parameter to `TRUE`.

For example:

```
BEGIN
DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL(
     username => 'adb_user',
     grant_option => TRUE);
END;
/
```

After you run this command, `adb_user` can enable resource principal for another user. For example, if you connect as `adb_user`, you can run the following command:

```
EXEC DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL(username => 'adb_user2');
```

See ENABLE_RESOURCE_PRINCIPAL Procedure for more information.

3. Verify that the resource principal credential is enabled.

   For example, as the ADMIN user query the view `DBA_CREDENTIALS`:

```
SELECT owner, credential_name FROM dba_credentials
        WHERE credential_name = 'OCI$RESOURCE_PRINCIPAL' AND owner =
'ADMIN';

OWNER  CREDENTIAL_NAME
-----  ----------------------
ADMIN  OCI$RESOURCE_PRINCIPAL
```

   For example, as a non-ADMIN user query the view `ALL_TAB_PRIVS`:

```
SELECT grantee, table_name, grantor, FROM ALL_TAB_PRIVS
          WHERE  grantee = 'ADB_USER';

GRANTEE   TABLE_NAME GRANTOR
--------- -----------------------------------
ADB_USER  OCI$RESOURCE_PRINCIPAL ADMIN
```

Enabling the resource principal on an Autonomous Database instance is one-time operation. You do not need to enable the resource principal again, unless you run `DBMS_CLOUD_ADMIN.DISABLE_RESOURCE_PRINCIPAL` to disable the resource principal.

Disable Resource Principal on Autonomous Database

Shows the steps to disable resource principal for all Autonomous Database users or for a specified user.

1. To disable resource principal for all users, as the ADMIN user, run the following command:

```
EXEC DBMS_CLOUD_ADMIN.DISABLE_RESOURCE_PRINCIPAL();
```

This removes the credential `OCI$RESOURCE_PRINCIPAL`.

2. Verify that the resource principal credential is disabled.

   For example:

```
SELECT owner, credential_name FROM dba_credentials
        WHERE credential_name = 'OCI$RESOURCE_PRINCIPAL' AND owner =
'ADMIN';

No rows selected
```

To remove access to the resource principal credential for a specified database user, include the `username` parameter. This denies the specified user access to the `OCI$RESOURCE_PRINCIPAL` credential.

For example:

```
EXEC DBMS_CLOUD_ADMIN.DISABLE_RESOURCE_PRINCIPAL(username => 'ADB_USER');
```

See DISABLE_RESOURCE_PRINCIPAL Procedure for more information.

Use Resource Principal with DBMS_CLOUD

When you specify resource principal credentials with `DBMS_CLOUD` calls, the database authenticates the Oracle Cloud Infrastructure requests for you and the database provides the credentials to access Oracle Cloud Infrastructure resources.

If you have not already done so, perform the required prerequisite steps:

- Access to Oracle Cloud Infrastructure resources depends on the dynamic group rules and the policies you set in Oracle Cloud Infrastructure policies and dynamic groups. See Perform Prerequisites to Use Resource Principal with Autonomous Database for more information.
- After you define the dynamic group and policies, enable the ADMIN schema or another schema to use a resource principal. See Enable Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

To use a `DBMS_CLOUD` procedure with resource principal credentials:

- Use a `DBMS_CLOUD` procedure or function and specify `OCI$RESOURCE_PRINCIPAL` as the credential name.

  For example, you can access Oracle Cloud Infrastructure Object Storage using a resource principal:

```
CREATE TABLE CHANNELS
   (channel_id CHAR(1),
    channel_desc VARCHAR2(20),
    channel_class VARCHAR2(20)
   );
/

BEGIN
 DBMS_CLOUD.COPY_DATA(
    table_name =>'CHANNELS',
    credential_name =>'OCI$RESOURCE_PRINCIPAL',
```

```
    file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/channels.txt',
    format => json_object('delimiter' value ',')
 );
END;
/
```

If you compare the steps required to access Object Storage as shown in Create Credentials and Copy Data into an Existing Table, notice that Step 1, creating credentials is not required when you use resource principal because you are using the system defined `OCI$RESOURCE_PRINCIPAL` credential.

Use Amazon Resource Names (ARNs) to Access AWS Resources

You can use Amazon Resource Names (ARNs) to access AWS resources with Autonomous Database.

- About Using Amazon Resource Names (ARNs) to Access AWS Resources
  When you use ARN role based authentication with Autonomous Database, you can securely access AWS resources without creating and saving credentials based on long-term AWS IAM access keys.
- Perform AWS Management Prerequisites to Use Amazon Resource Names (ARNs)
  Using the AWS Management Console or using the APIs, create an AWS user, role, policies, and trust relationship. You perform these steps before you use with `DBMS_CLOUD.CREATE_CREDENTIAL` to create a credential with an ARN parameter on Autonomous Database.
- Perform Autonomous Database Prerequisites to Use Amazon ARNs
  Prior to using an AWS resource with `DBMS_CLOUD.CREATE_CREDENTIAL` with an ARN parameter, the ADMIN user must enable ARN on the Autonomous Database instance.
- Create Credentials with ARN Parameters to Access AWS Resources
  After ARN usage is enabled for the Autonomous Database instance and the ARN is configured by the AWS administrator, on Autonomous Database you can create a credential object with ARN parameters.
- Update Credentials with ARN Parameters for AWS Resources
  The ARN credentials you use on Autonomous Database work with the AWS token service that enables you to use temporary role based credentials to access to AWS resources from Autonomous Database.

About Using Amazon Resource Names (ARNs) to Access AWS Resources

When you use ARN role based authentication with Autonomous Database, you can securely access AWS resources without creating and saving credentials based on long-term AWS IAM access keys.

For example, you may want to load data from an AWS S3 bucket into your Autonomous Database, perform some operation on the data, and then write the modified data back to the S3 bucket. You can do this without using an ARN if you have AWS user credentials to access the S3 bucket. However, using role-based ARNs to access AWS resources from Autonomous Database has the following benefits:

- You can create role-based access, with different policies for different users or schemas that need access to AWS resources from an Autonomous Database instance. This allows you to set a policy to limit access to AWS resources by role. For example, setting a policy limiting to read-only access, by role, to an S3 bucket.

- ARN based credentials provide better security as you do not need to provide long-term AWS user credentials in code to access AWS resources. Autonomous Database manages the temporary credentials generated from the AWS Assume Role Operation.

Steps to Configure ARN Usage with Autonomous Database

Before creating a credential using an ARN in Autonomous Database, in AWS, your account administrator must define a policy that allows you to access AWS resources, such as an S3 bucket. By default, ARN credential services are not enabled on Autonomous Database. The ADMIN user enables ARN credentials for the necessary user which allows them to create and use ARN credentials on the Autonomous Database instance.

In AWS, the role ARN is the identifier for the provided access and can be viewed on the AWS console. For added security, when the AWS administrator configures the role, policies, and trust relationship for the AWS account, they must also configure an External ID in the role's trust relationship.

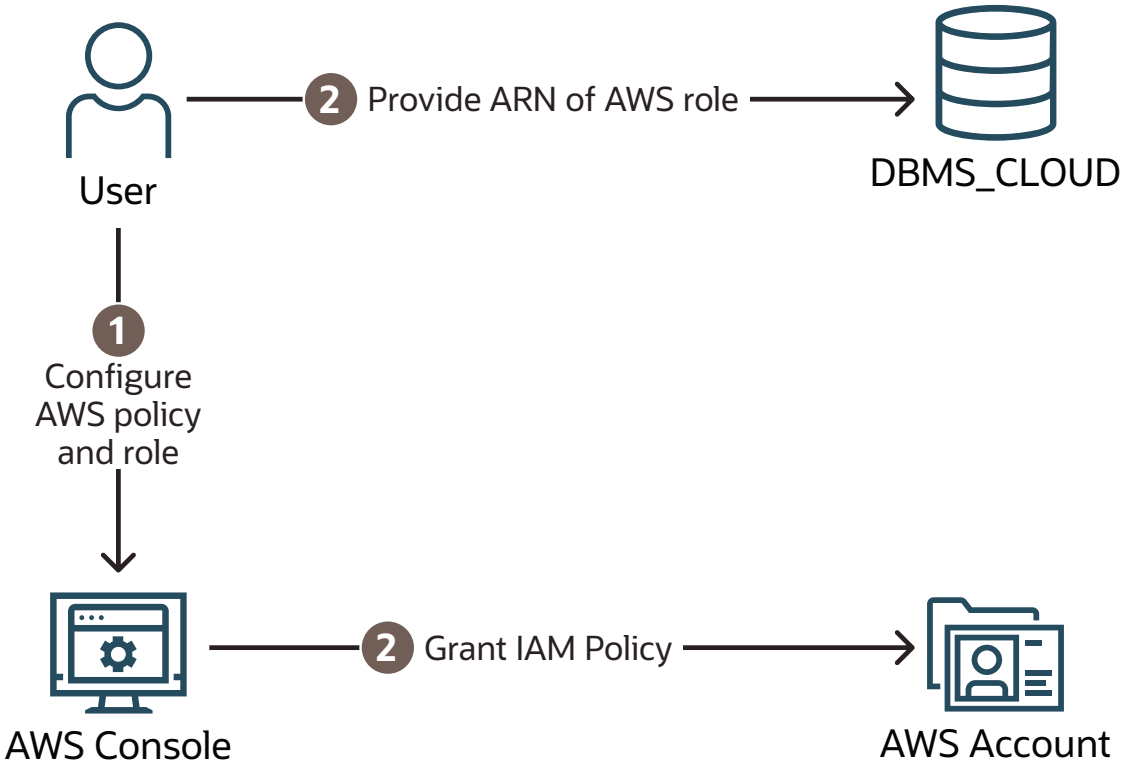
The External ID provides additional protection for assuming roles. The AWS administrator configures configure the External ID as one of the following, based on the Autonomous Database instance:

- The compartment OCID
- The database OCID
- The tenancy OCID

On AWS, the role can only be assumed by trusted users that are identified by the External ID included in the request URL, where the supplied External ID in the request matches the External ID configured in the role's trust relationship.

> **Note:**
>
> Setting the External ID is required for security.

The following figure outlines the configuration steps:



Steps to Use ARNs with DBMS_CLOUD

Each AWS resource has its own identity, and the resource authenticates with the Autonomous Database instance using a `DBMS_CLOUD` credential that you create with parameters that identify the ARN. Autonomous Database creates and secures the principal credentials you use to access AWS resources.

To create a credential with ARN parameters to access AWS resources:

1. Perform prerequisite steps in the AWS Account: In your AWS account, from the AWS Management Console or using the CLI, create the roles and policies for the ARN that you use with Autonomous Database and update the trust relationship for the role. The Oracle user ARN is configured when the trust relationship for the role is updated.

   See Perform AWS Management Prerequisites to Use Amazon Resource Names (ARNs) for more information.

2. Perform prerequisite steps on Autonomous Database: On Autonomous Database you must enable the ADMIN user or another user to use credentials with ARN parameters to access AWS resources.

   See Perform Autonomous Database Prerequisites to Use Amazon ARNs for more information.

3. Create credentials with `DBMS_CLOUD.CREATE_CREDENTIAL` and supply the parameters that identify an AWS role. Using the credential object, Autonomous Database can access AWS resources as specified in the policies defined for the role in the AWS account.

   See Create Credentials with ARN Parameters to Access AWS Resources for details on these steps.

4. Use the credential object you created in the previous step with a `DBMS_CLOUD` procedure or function that takes a credential parameter, such as `DBMS_CLOUD.COPY_DATA` or `DBMS_CLOUD.LIST_OBJECTS`.

Perform AWS Management Prerequisites to Use Amazon Resource Names (ARNs)

Using the AWS Management Console or using the APIs, create an AWS user, role, policies, and trust relationship. You perform these steps before you use with `DBMS_CLOUD.CREATE_CREDENTIAL` to create a credential with an ARN parameter on Autonomous Database.

To use an ARN to access AWS resources your AWS administrator defines the policies and a principal that allows you to access AWS resources. For example, while using Autonomous Database you might want to access data from an S3 bucket, perform some operation on the data, and then write the modified data back to the S3 bucket.

> **Note:**
>
> Depending on your existing AWS configuration and the External ID you use, you do not need to create a new role and policy for each Autonomous Database instance. If you already have an AWS role containing the necessary policy to access a resource, for example to access S3 cloud storage, you can modify the trust relationship to include the details in Step 3. Likewise, if you already have a role with the necessary trust relationship, you can use that role to access all of your databases in an OCI compartment or tenancy if you use an external ID that specifies the compartment OCID or tenancy OCID.

From the AWS Management Console or using the APIs, an AWS administrator performs the following steps:

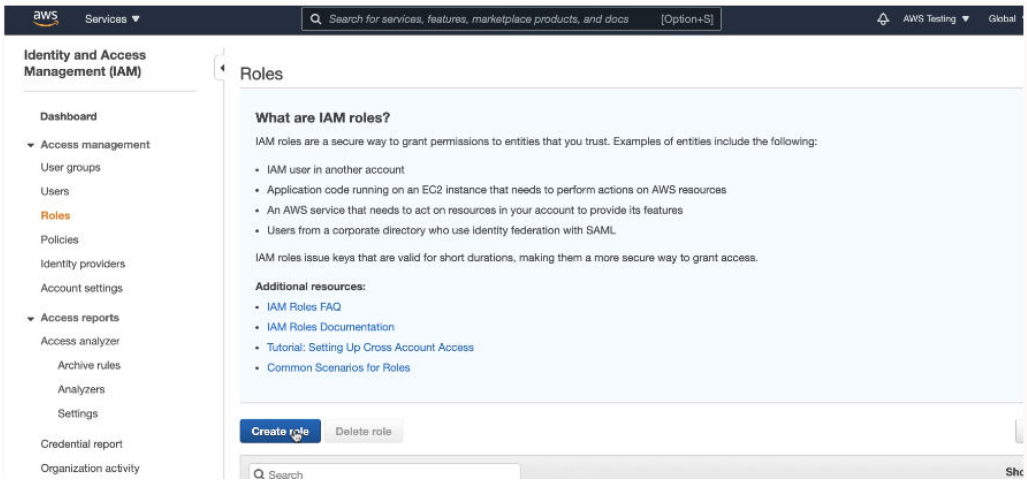
1. Create a policy. In the policy you specify permissions for accessing AWS resources such as S3 buckets.

   See Creating an IAM policy to access Amazon S3 resources for more information.

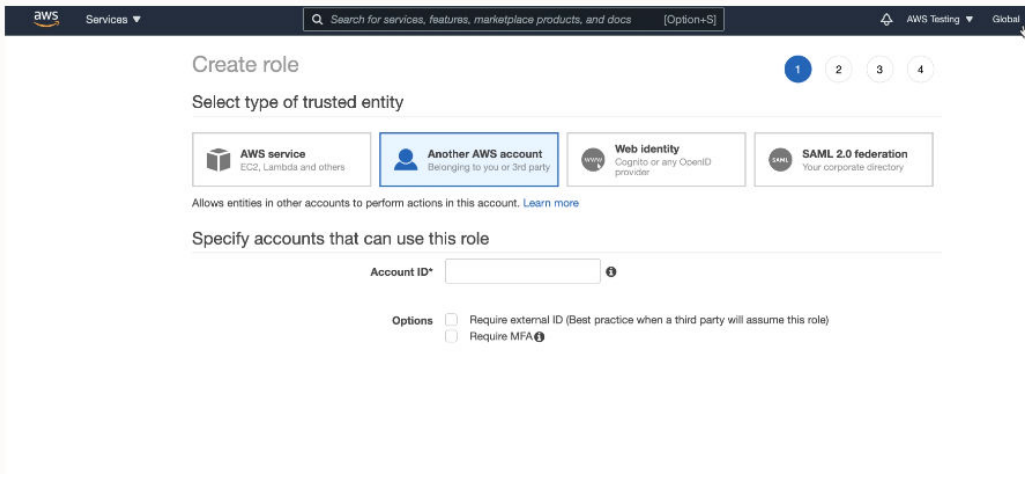
2. Create a role and attach the policy to the role.

   a. Access the AWS Management Console and choose Identity and Access Management (IAM).

   b. Click **Create role**.



   c. Select **Another AWS account**.



d. Enter your **Account ID**.

   You use this as a temporary value. Later you replace this with the Account ID you use to access AWS resources.

e. In the **Options** area select **Require external ID** and enter a temporary external ID, such as `0000`. Later you replace this external ID with a valid value.

f. Click **Next Permissions** to attach the Policies you created in Step 1 or other policies you want to apply to the role.

g. Click **Next Tags** and apply or create tags as needed for the role.

h. Click **Next Review** and add a **Role Name** and **Role Description**.

i. Click **Create Role**.

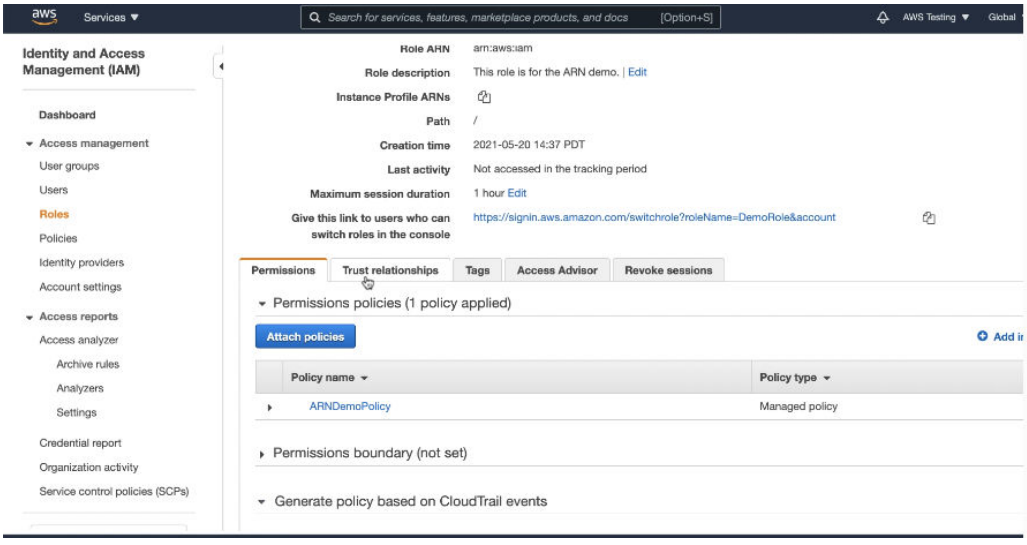
You use the role's ARN with `DBMS_CLOUD.CREATE_CREDENTIAL` to create credential objects with ARN parameters to access AWS resources.

See Creating a role to delegate permissions to an IAM user for more information.

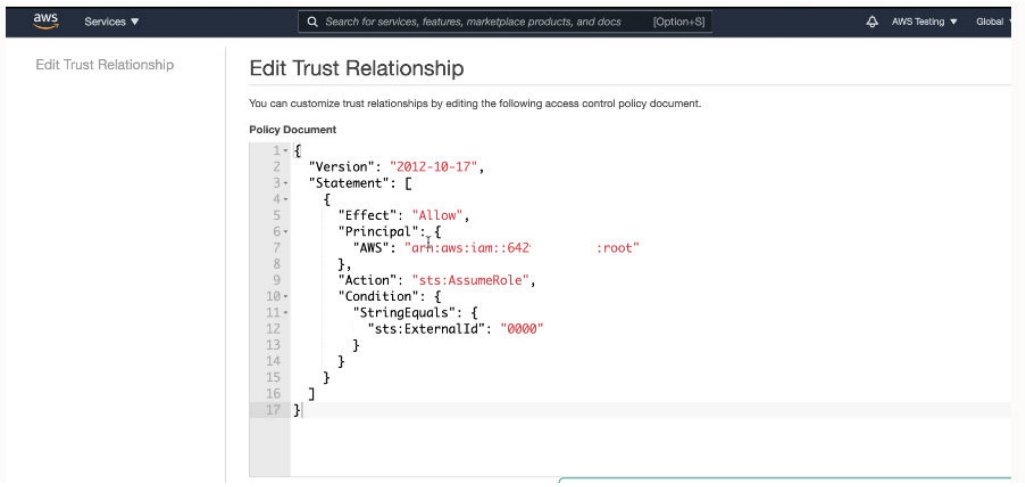
3. Specify a Trust Relationship for the role.

   a. From the Roles list, under **Role name**, select the role you created.

   b. On the roles Summary page for the selected role, select the **Trust relationships** tab.



   c. In the trust relationship, click **Edit trust relationship**.



d. Edit the trust relationship to specify the Principal parameter **AWS**.

   This AWS user ARN is available in the CLOUD_INTEGRATIONS view. See Perform Autonomous Database Prerequisites to Use Amazon ARNs for more information.

e. Edit the trust relationship to specify the External ID.

   On Autonomous Database when you create an AWS ARN credential with `DBMS_CLOUD.CREATE_CREDENTIAL` or when you enable AWS ARN with `DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH`, by default the `external_id_type` parameter value is `database_ocid`. Optionally you can set the `external_id_type` value to one of the supported values: `database_ocid`, `compartment_ocid`, or `tenant_ocid`.

   When you use the database OCID as the External ID, the policy's trust relationship only trusts the Autonomous Database instance specified with the OCID. If you use a compartment OCID, the policy's trust relationship trusts all the Autonomous Database instances in the compartment and you can use the same role ARN to grant access to AWS resources to any Autonomous Database in the specified compartment. Likewise, if you use the tenancy OCID, you can use the same role ARN to grant access to AWS resources to any Autonomous Database in the specified tenancy.

   Previously in Step 2 you set the trust relationship External ID to the temporary value `0000`.

   On AWS you configure the trust relationship External ID value to match one of the following:

   - When the `external_id_type` type is `database_ocid`, on AWS you configure the role's trust relationship External ID to be the Database OCID.

     The Database OCID is available by running the following query:

     ```
     SELECT cloud_identity FROM v$pdbs;
     ```

     See Obtain Tenancy Details for more information.

   - When the `external_id_type` type is `compartment_ocid`, on AWS you configure the role's trust relationship External ID to be the Compartment OCID.

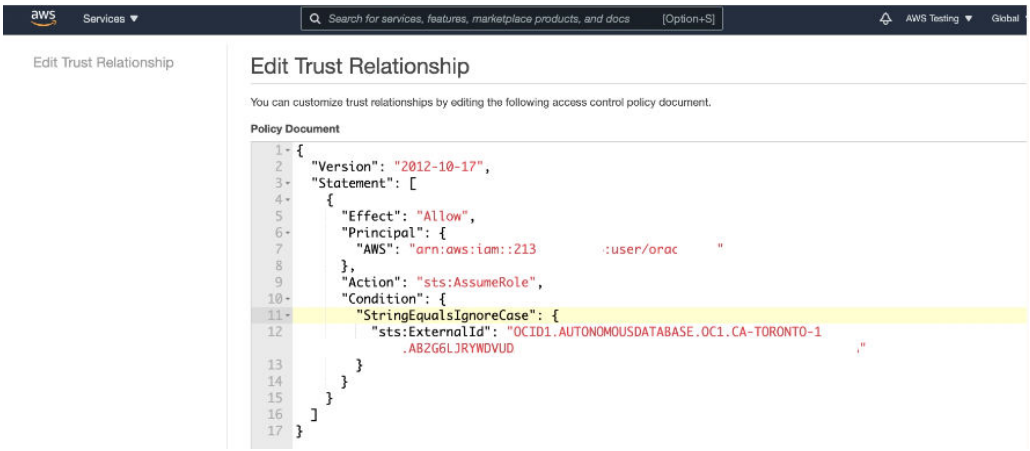
     The Compartment OCID is available on the Compartment details page from the Oracle Cloud Infrastructure Console. To find the Compartment details page, from the Oracle Cloud Infrastructure left navigation menu click **Identity & and Security**

and then select **Compartments**. Select the compartment that contains the Autonomous Database instance to see the Compartment ID.

- When the `external_id_type` type is `tenant_ocid`, on AWS you configure the role's trust relationship External ID to be the Tenancy OCID.

  The Tenancy OCID is available on the Tenancy details page from the Oracle Cloud Infrastructure Console. To find the Tenancy details page, from the Oracle Cloud Infrastructure left navigation menu click **Governance & Administration** and then select **Tenancy Details**. The Tenancy Information tab shows the Tenancy OCID.

- When you set the value for ExternalID, by default the OCID value must be in upper case. If you want to supply the OCID in lower case, set the condition "`StringEqualsIgnoreCase`" instead of "`StringEquals`" in the JSON when you edit the trust relationship.



See How to use trust policies with IAM role and How to use an external ID when granting access to your AWS resources to a third party for more information.

After the ARN role configuration is finished, you can enable ARN on the instance. See Perform Autonomous Database Prerequisites to Use Amazon ARNs for more information.

Perform Autonomous Database Prerequisites to Use Amazon ARNs

Prior to using an AWS resource with `DBMS_CLOUD.CREATE_CREDENTIAL` with an ARN parameter, the ADMIN user must enable ARN on the Autonomous Database instance.

By default, ARN credential services are not enabled on Autonomous Database. The ADMIN user runs the procedure `DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH` to enable the ADMIN user or other users to create credentials with ARN parameters.

1. Enable the use of ARN credentials on the Autonomous Database instance.

```
BEGIN
    DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
        username => 'adb_user',
        params => JSON_OBJECT(
                'aws_role_arn' value 'arn:aws:iam::123456:role/
AWS_ROLE_ARN'));
END;
/
```

If you want the specified user to have privileges to enable ARN credentials for other users, set the `params` parameter `grant_option` to `TRUE`.

For example:

```
BEGIN
DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
     username => 'adb_user',
     params => JSON_OBJECT(
                      'aws_role_arn' value 'arn:aws:iam::123456:role/
AWS_ROLE_ARN',
                      'grant_option' value TRUE ));
END;
/
```

After you run this command, `adb_user` has privileges to enable ARN credentials for other users.

For example, if you connect as `adb_user`, you can run the following command:

```
BEGIN
    DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
        username => 'adb_user2');
END;
/
```

See ENABLE_PRINCIPAL_AUTH Procedure for more information.

2. Query the `CLOUD_INTEGRATIONS` view to obtain Oracle's AWS user ARN.

```
SELECT param_value FROM CLOUD_INTEGRATIONS
        WHERE param_name = 'aws_user_arn';

PARAM_VALUE
-----------------------------------------
arn:aws:iam::account-ID:user/username
```

The view `CLOUD_INTEGRATIONS` is available to the ADMIN user or to a user with `DWROLE` privileges.

The AWS administrator uses the `aws_user_arn` value when configuring the AWS role's trust relationship with the role and policies on the AWS system. Providing this value grants permission on the AWS side for `DBMS_CLOUD` to access AWS resources.

After you enable ARN on the Autonomous Database instance by running `DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH`, the credential named `AWS$ARN` is available to use with any `DBMS_CLOUD` API that takes a credential as the input. Except for the credential named `AWS$ARN`, you can also create additional credentials with ARN parameters to access AWS resources. See Create Credentials with ARN Parameters to Access AWS Resources for more information.

Create Credentials with ARN Parameters to Access AWS Resources

After ARN usage is enabled for the Autonomous Database instance and the ARN is configured by the AWS administrator, on Autonomous Database you can create a credential object with ARN parameters.

Autonomous Database creates and secures the principal credentials you use to access the Amazon resources when you supply the credential object with `DBMS_CLOUD` procedures and functions.

To use Amazon resources with Autonomous Database, do the following:

1. Create credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL` with the `params` parameter to specify the ARN value. For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_ARN',
    params =>
        JSON_OBJECT('aws_role_arn' value 'arn:aws:iam::123456:role/
AWS_ROLE_ARN',
                    'external_id_type' value 'database_ocid')
  );
END;
/
```

   This operation creates the credentials in the database in an encrypted format. You can use any name for the credential name.

   For detailed information about the parameters, see CREATE_CREDENTIAL Procedure.

2. Use a `DBMS_CLOUD` procedure to access an Amazon resource with the ARN credentials.

   For example, use `DBMS_CLOUD.LIST_OBJECTS`.

```
SELECT object_name FROM DBMS_CLOUD.LIST_OBJECTS(
           credential_name => 'DEF_CRED_ARN',
           location_uri    => 'https://my-bucket.s3.us-
west-2.amazonaws.com/');
```

Update Credentials with ARN Parameters for AWS Resources

The ARN credentials you use on Autonomous Database work with the AWS token service that enables you to use temporary role based credentials to access to AWS resources from Autonomous Database.

When an AWS Administrator revokes the policies, roles, or trust relationship, you need to either update the credentials or create new credentials to access the AWS resources.

Perform the following steps to update credentials:

1. Use `DBMS_CLOUD.UPDATE_CREDENTIAL` to update an ARN based credential to supply a new ARN value.

```
BEGIN
  DBMS_CLOUD.UPDATE_CREDENTIAL(
     credential_name => 'DEF_CRED_ARN',
```

```
    attribute => 'aws_role_arn',
    value => 'new_ARN_value');
END;
/
```

This updates the `aws_role_arn` attribute to the new value *`new_ARN_value`* for the credential named DEF_CRED_ARN.

2. Use `DBMS_CLOUD.UPDATE_CREDENTIAL` to update an ARN based credential to update the attribute `external_id_type` value.

```
BEGIN
  DBMS_CLOUD.UPDATE_CREDENTIAL(
    credential_name => 'DEF_CRED_ARN',
    attribute => 'external_id_type',
    value => 'compartment_ocid');
END;
/
```

This updates the `external_id_type` attribute value to the value `compartment_ocid`.

See UPDATE_CREDENTIAL Procedure and CREATE_CREDENTIAL Procedure for more information.

Use Azure Service Principal to Access Azure Resources

You can use an Azure service principal with Autonomous Database to access Azure resources without having to create and store your own credential objects in the database.

- Enable Azure Service Principal
  Enable Azure service principal authentication to allow Autonomous Database to access Azure services without providing long-term credentials.
- Provide Azure Application Consent and Assign Roles
  To access Azure resources from Autonomous Database with Azure service principal authentication you must consent the Azure application and assign roles to allow access to your Azure resources.
- Use Azure Service Principal with DBMS_CLOUD
  When you make `DBMS_CLOUD` calls to access Azure resources and specify the credential name as `AZURE$PA`, the authentication on the Azure side happens using the Azure service principal.
- Disable Azure Service Principal
  To disable access to Azure resources from Autonomous Database with Azure service principal, use `DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH`.
- Notes for Azure Service Principal
  Notes for using Azure service principal.

Enable Azure Service Principal

Enable Azure service principal authentication to allow Autonomous Database to access Azure services without providing long-term credentials.

> **Note:**
>
> To use Autonomous Database with Azure service principal authentication you need a Microsoft Azure account. See Microsoft Azure for details.

To enable Azure service principal authentication on Autonomous Database:

1. Obtain your Microsoft Azure Active Directory tenant ID.

   See How to find your Azure Active Directory tenant ID for more information.

2. Enable Azure service principal with `DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH`.

   For example:

```
BEGIN
    DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
        provider => 'AZURE',
        username => 'adb_user',
        params   => JSON_OBJECT('azure_tenantid' value 'azure_tenantID'));
END;
/
```

   This enables Azure service principal authentication and creates an Azure application on Autonomous Database.

   If you want the specified user to have privileges to enable Azure service principal for other users, set the `params` parameter `grant_option` to `TRUE`.

   For example:

```
BEGIN
    DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
        provider => 'AZURE',
        username => 'adb_user',
        params   => JSON_OBJECT('grant_option' value TRUE,
                                'azure_tenantid' value 'azure_tenantID'));
END;
/
```

   After you run this command, `adb_user` can enable Azure service principal for another user. For example, if you connect as `adb_user`, you can run the following command:

```
BEGIN
    DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
        provider => 'AZURE',
        username => 'adb_user2');
END;
/
```

See ENABLE_PRINCIPAL_AUTH Procedure for more information.

Provide Azure Application Consent and Assign Roles

To access Azure resources from Autonomous Database with Azure service principal authentication you must consent the Azure application and assign roles to allow access to your Azure resources.

To provide Azure application consent and assign roles, perform the following steps:

1. On Autonomous Database query `CLOUD_INTEGRATIONS`.

   For example:

   ```
   SELECT param_name, param_value FROM CLOUD_INTEGRATIONS;

   PARAM_NAME        PARAM_VALUE
   ---------------
   --------------------------------------------------------------------------------
   ----------------------------------------------------------------
   azure_tenantid    29981886-6fb3-44e3-82ab-d870b0e8e7eb
   azure_consent_url https://login.microsoftonline.com/f8cdef31-91255a/oauth2/
   v2.0/authorize?
   client_id=d66f1b5-1250d5445c0b&response_type=code&scope=User.read
   azure_app_name
   ADBS_APP_OCID1.AUTONOMOUSDATABASE.REGION1.SEA.ANZWKLJSZLYNB3AAWLYL3JVC4ICEX
   LB3ZG6WTCX735JSSY2NRHOBU4DZOOVA
   ```

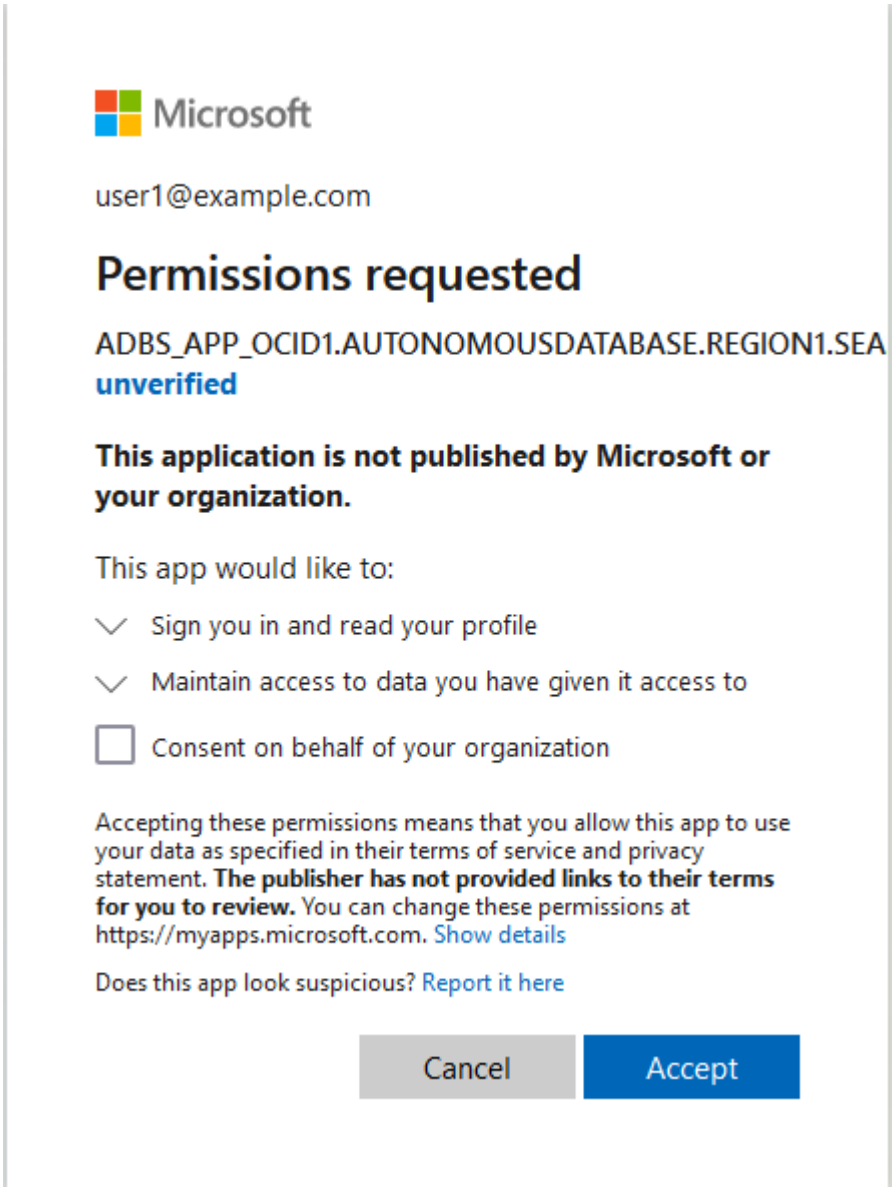
   The view `CLOUD_INTEGRATIONS` is available to the `ADMIN` user or to a user with `DWROLE` role.

2. In a browser, open the Azure consent URL, specified by the `azure_consent_url` parameter.

   For example, copy and enter the URL in your browser:

   ```
   https://login.microsoftonline.com/f8cdef31-91255a/oauth2/v2.0/authorize?
   client_id=d66f1b5-1250d5445c0b&response_type=code&scope=User.read
   ```

   The **Permissions requested** page opens and shows a consent request, similar to the following:



3. To provide consent click **Accept**.
4. On the Microsoft Azure console, assign the roles you want to grant to allow access to the specified Azure resources.

   For example, if you want to access Azure Blob Storage from Autonomous Database, assign roles so that the Azure application (the service principal) has access to Azure Blob Storage.

> **Note:**
>
> To work with Azure Blob Storage, you need an Azure storage account. If you do not have an Azure storage account, create a storage account. See Create a storage account for more information.

   a. On the Microsoft Azure console, under **Azure services**, select **Storage accounts**.

b. Under **Storage accounts**, click the storage account you want to grant service principal access to.

c. On the left, click **Access Control (IAM)**.

d. From the top area, click **+ Add → Add role assignment**.

e. In the search area enter text to narrow the list of roles that you see. For example, enter **Storage Blob** to show the available roles that contain **Storage Blob**.

f. Select one or more roles as appropriate for the access you want to grant. For example, select **Storage Blob Data Contributor**.

g. Click **Next**.

h. In the **Add role assignment**, under **Members** click **+ Select members**.

i. Under **Select members**, in the select field, enter the `azure_app_name` listed in Step 1 (the `param_value` column of the `CLOUD_INTEGRATIONS` view).

j. Select the application.

For example, click `ADBS_APP_OCID1.AUTONOMOUSDATABASE.REGION1.SEA.ANZWKLJSZLYNB3AAWLYL3JVC4ICEXLB3ZG6WTCX735JSSY2NRHOBU4DZOOVA`

k. Click **Select**.

l. Click **Review + assign**.

5. Click **Review + Assign** again.

After assigning a role you need to wait, as role assignments may take up to five minutes to propagate in Azure.

This example shows steps to grant roles for accessing Azure Blob Storage. If you want to provide access for other Azure services you need to perform equivalent steps for the additional Azure services to allow the Azure application (the service principal) to access the Azure service.

Use Azure Service Principal with DBMS_CLOUD

When you make `DBMS_CLOUD` calls to access Azure resources and specify the credential name as `AZURE$PA`, the authentication on the Azure side happens using the Azure service principal.

If you have not already done so, perform the required prerequisite steps:

- Enable the ADMIN schema or another schema to use Azure service principal authentication. See Enable Azure Service Principal for more information.
- Consent the application and perform the Azure role assignment grants. See Provide Azure Application Consent and Assign Roles for more information.

To use a `DBMS_CLOUD` procedure or function with Azure service principal, specify `AZURE$PA` as the credential name. For example, you can access Azure Blob Storage using Azure service principal credentials as follows:

```
SELECT * FROM DBMS_CLOUD.LIST_OBJECTS('AZURE$PA', 'https://
treedata.blob.core.windows.net/treetypes/');

OBJECT_NAME BYTES CHECKSUM                CREATED
LAST_MODIFIED
----------- ----- ----------------------- --------------------
--------------------
```

```
trees.txt      58 aCB1qMOPVobDLCXG+2fcvg== 2022-04-07T23:03:01Z
2022-04-07T23:03:01Z
```

If you compare the steps required to access object storage, as shown in Create Credentials and Copy Data into an Existing Table, notice that Step 1, creating credentials is not required because you are using an Azure service principal called `AZURE$PA`.

Disable Azure Service Principal

To disable access to Azure resources from Autonomous Database with Azure service principal, use `DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH`.

To disable Azure service principal on Autonomous Database:

```
BEGIN
    DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH(
        provider => 'AZURE',
        username => 'adb_user');
END;
/
```

When the `provider` value is `AZURE` and the `username` is a user other than the `ADMIN` user, the procedure revokes the privileges from the specified user. In this case, the `ADMIN` user and other users can continue to use `ADMIN.AZURE$PA` and the application that is created for the Autonomous Database instance remains on the instance.

When the `provider` value is `AZURE` and the `username` is `ADMIN`, the procedure disables Azure service principal based authentication and deletes the Azure service principal application on the Autonomous Database instance. In this case, if you want to enable Azure service principal you must perform all the steps required to use Azure service principal again, including the following:

- Enable the `ADMIN` schema or another schema to use Azure service principal authentication. See Enable Azure Service Principal for more information.
- Consent the application and perform the Azure role assignment grants. See Provide Azure Application Consent and Assign Roles for more information.

See DISABLE_PRINCIPAL_AUTH Procedure for more information.

Notes for Azure Service Principal

Notes for using Azure service principal.

- **Cloning an Autonomous Database instance with Azure service principal**: When you clone an instance with Azure service principal enabled, the Azure service principal configuration is not carried over to the clone. Perform the steps to enable Azure service principal on the clone if you want to enable Azure service principal on a cloned instance.

Use Google Service Account to Access Google Cloud Platform Resources

You can use a Google service account to access Google Cloud Platform (GCP) resources from an Autonomous Database instance.

- About Using a Google Service Account to Access Google Cloud Resources
  When you use Google service account based authentication with Autonomous Database, an application can securely access Google Cloud Platform (GCP) resources without

creating and saving credentials based on long-term IAM access keys for the GCP resources.

- Enable Google Service Account and Find the GCP Service Account Name
  Prior to using a Google Cloud Platform (GCP) resource with a Google service account you need to enable GCP access for your Autonomous Database instance.
- Assign Roles to the Google Service Account and Provide Access for GCP Resources
  To use Google Cloud Platform (GCP) resources from an Autonomous Database instance, you or a Google Cloud Administrator must assign roles and privileges to the Google service account that your application accesses. In addition to assigning roles for the Google service account, for any GCP resources you want to use a Google Cloud administrator must add Google IAM principals.
- Use Google Service Account with DBMS_CLOUD
  When you make `DBMS_CLOUD` calls to access Google Cloud Platform (GCP) resources and specify the credential name as `GCP$PA`, the authentication on the Google Cloud Platform side happens using a Google service account.
- Disable Google Service Account
  To disable Google service account access to Google Cloud Platform (GCP) resources, use `DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH`.
- Google Service Account Notes
  Notes for using Google service account.

About Using a Google Service Account to Access Google Cloud Resources

When you use Google service account based authentication with Autonomous Database, an application can securely access Google Cloud Platform (GCP) resources without creating and saving credentials based on long-term IAM access keys for the GCP resources.

A Google service account is a special kind of GCP account used by an application. You can use a Google service account to make authorized GCP REST API calls from an application (after the service account is given access permissions through IAM role configuration). When an application makes calls with GCP service account based authentication, the initial call generates a temporary access token through OAuth2.0. The OAuth2.0 access token is valid for one hour. Subsequent requests within the hour use the OAuth2.0 access token to make authorized GCP REST API calls.

For example, you may want to load data from Google Cloud Storage into your Autonomous Database, perform some operation on the data, and then write the modified data back to Google Cloud Storage. You can do this without using a service account if you have GCP user credentials to access Google Cloud Storage. However, using a role-based Google service account to access GCP resources from Autonomous Database has the following benefits:

- You can create role-based access, with different policies for different users or schemas that need access to GCP resources from an Autonomous Database instance. This allows you to set a policy to limit access to resources by role. For example, setting a policy that is limited to read-only access, by role, to a Google Cloud Storage bucket.
- Google service account based credentials provide better security, as you do not need to provide long-term user credentials in code when your application accesses GCP resources. Autonomous Database manages the temporary credentials for the Google service account and does not need to store GCP resource user credentials in the database.

See Service accounts for information on Google service accounts.

Enable Google Service Account and Find the GCP Service Account Name

Prior to using a Google Cloud Platform (GCP) resource with a Google service account you need to enable GCP access for your Autonomous Database instance.

1. Enable Google service account authentication with `DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH`.

   For example, to enable Google service account authentication for the `ADMIN` user:

   ```
   BEGIN
       DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
           provider => 'GCP' );
   END;
   /
   ```

   Enable Google service account authentication for a non-`ADMIN` user, `adb_user` as follows:

   ```
   BEGIN
       DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
           provider => 'GCP',
           username => 'adb_user');
   END;
   /
   ```

   If you want the specified user to have privileges to enable Google service account authentication for other users, set the `params` parameter `grant_option` to `TRUE`.

   ```
   BEGIN
       DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
           provider => 'GCP',
           username => 'adb_user',
           params   => JSON_OBJECT('grant_option' value TRUE));
   END;
   /
   ```

   After you run `DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH` with `grant_option` set to `TRUE`, `adb_user` can enable Google service account authentication for another user. For example, if you connect as `adb_user`, you can run the following command to enable GCP service account access for `adb_user2`:

   ```
   BEGIN
       DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
           provider => 'GCP',
           username => 'adb_user2');
   END;
   /
   ```

2. When `DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH` runs it creates a Google service account. Query `CLOUD_INTEGRATIONS` to obtain the service account details for your Autonomous Database instance.

```
SELECT * FROM CLOUD_INTEGRATIONS WHERE param_name = 'gcp_service_account';

PARAM_NAME           PARAM_VALUE
-------------------------------------------------------------------------
gcp_service_account  GCP-SA-22222-32222@gcp-example.iam.gserviceaccount.com
```

3. Note the `gcp_service_account` parameter value as you must supply this value when you configure GCP resources.

See ENABLE_PRINCIPAL_AUTH Procedure for more information.

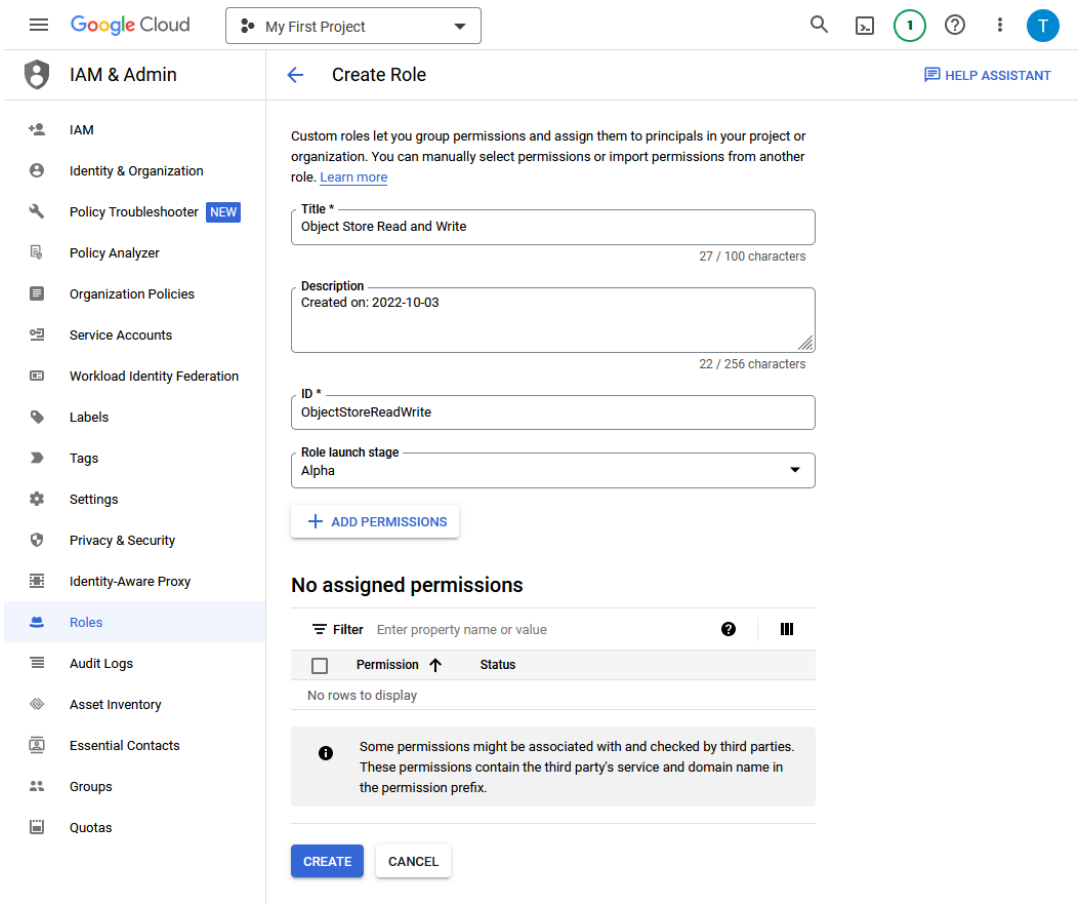
Assign Roles to the Google Service Account and Provide Access for GCP Resources

To use Google Cloud Platform (GCP) resources from an Autonomous Database instance, you or a Google Cloud Administrator must assign roles and privileges to the Google service account that your application accesses. In addition to assigning roles for the Google service account, for any GCP resources you want to use a Google Cloud administrator must add Google IAM principals.

As a prerequisite, first enable the Google service account on your Autonomous Database instance. See Enable Google Service Account and Find the GCP Service Account Name for more information.

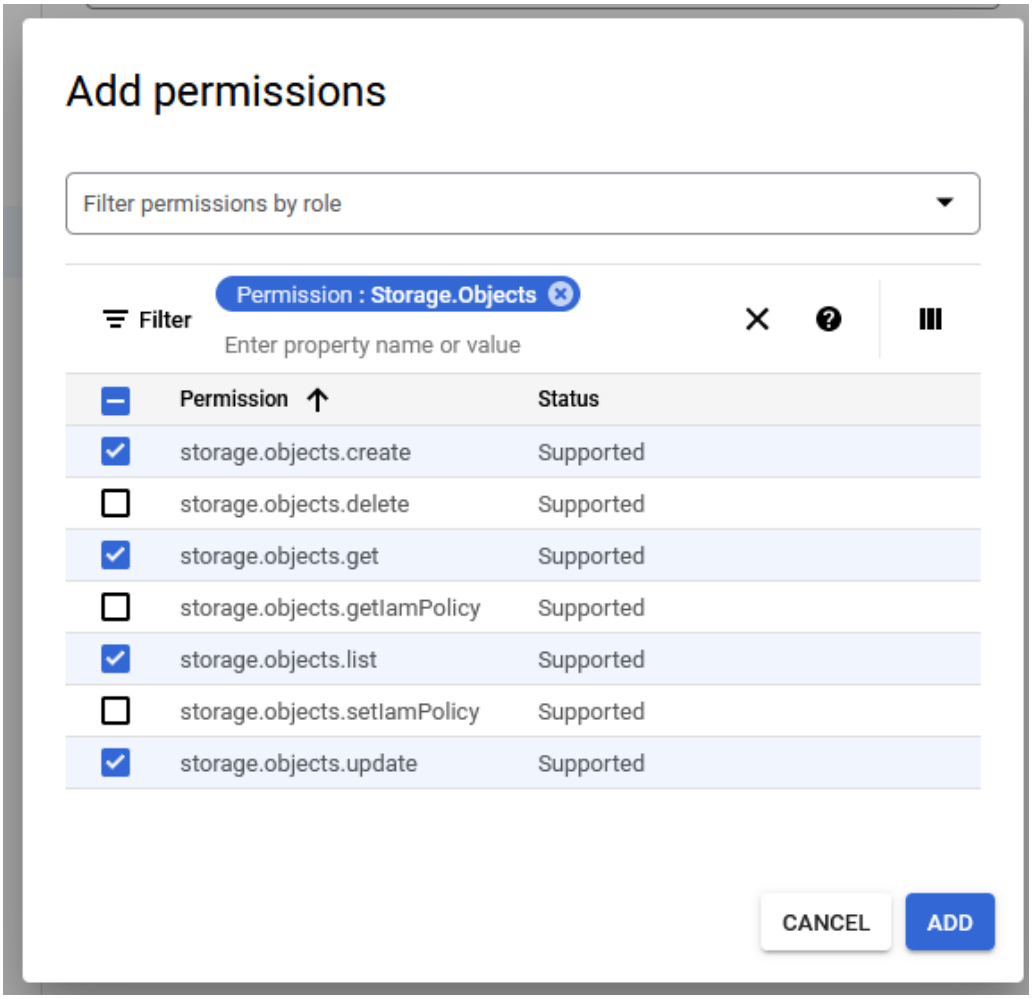
1. Open the Google Cloud Console for your account.
2. Create roles with the specified permissions.
   a. From the navigation menu, select **IAM & Admin**.
   b. From the IAM & Admin navigator, select **Roles**.
   c. On the Roles page, click More Actions and select **+ CREATE ROLE**.

   For example, you can create a role **Object Store Read and Write** to control the use of an Object Store bucket.



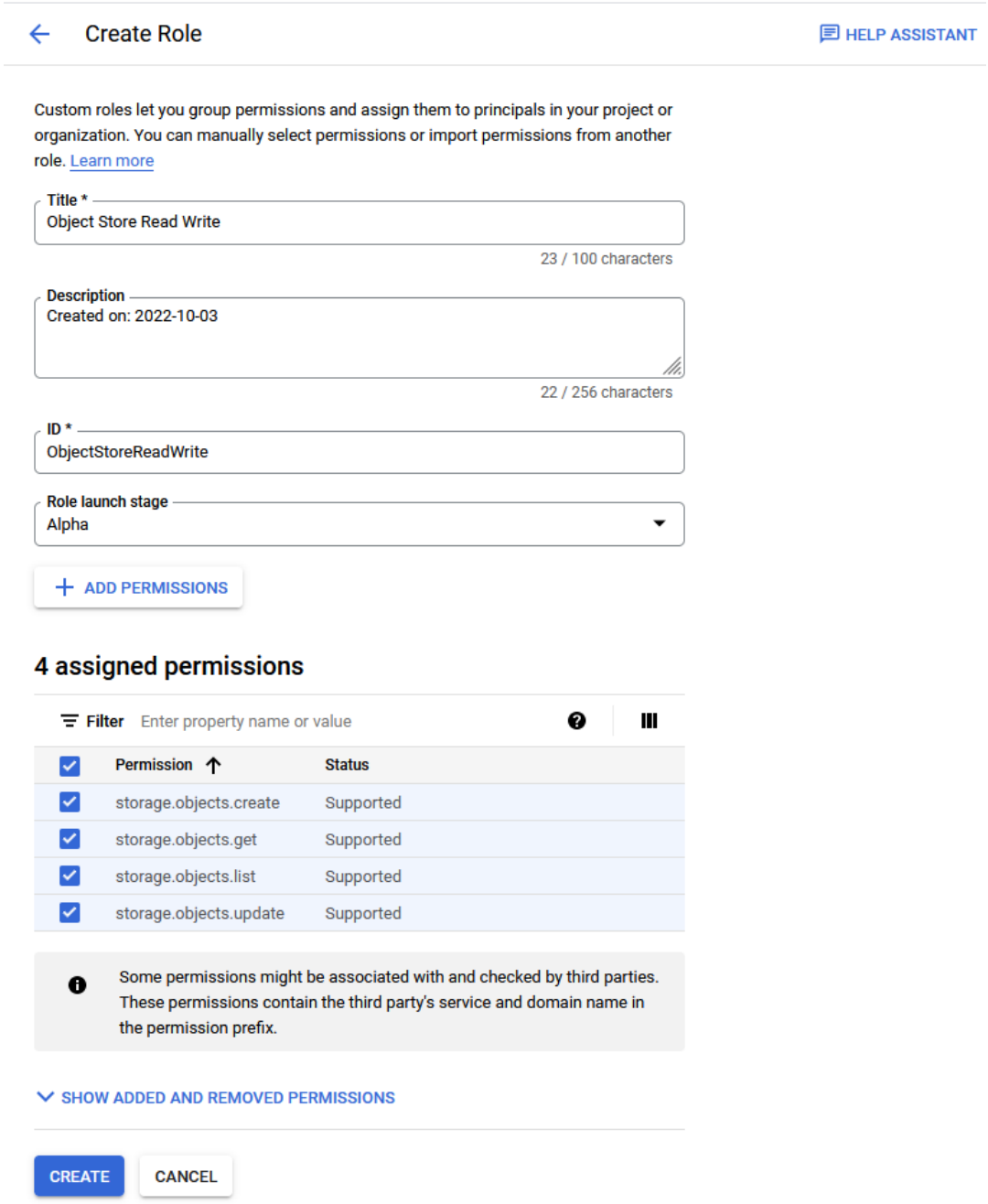
3. On the Create Role page, click **+ ADD PERMISSIONS**.
   a. Select filters to limit the list of permissions.

      For example, enter the filter **Permission: Storage.Objects** to show only the Object Store permissions.



b. On the Add permissions dialog, click **ADD**.

4. On the Create Role page, click **CREATE**.



5. Add roles and principals for the resource you want to access.

   For example, if you want to access Google Cloud Storage using the role you just created, **Object Store Read Write**:

   a. From the navigator, select **Cloud Storage** and select **Buckets**.

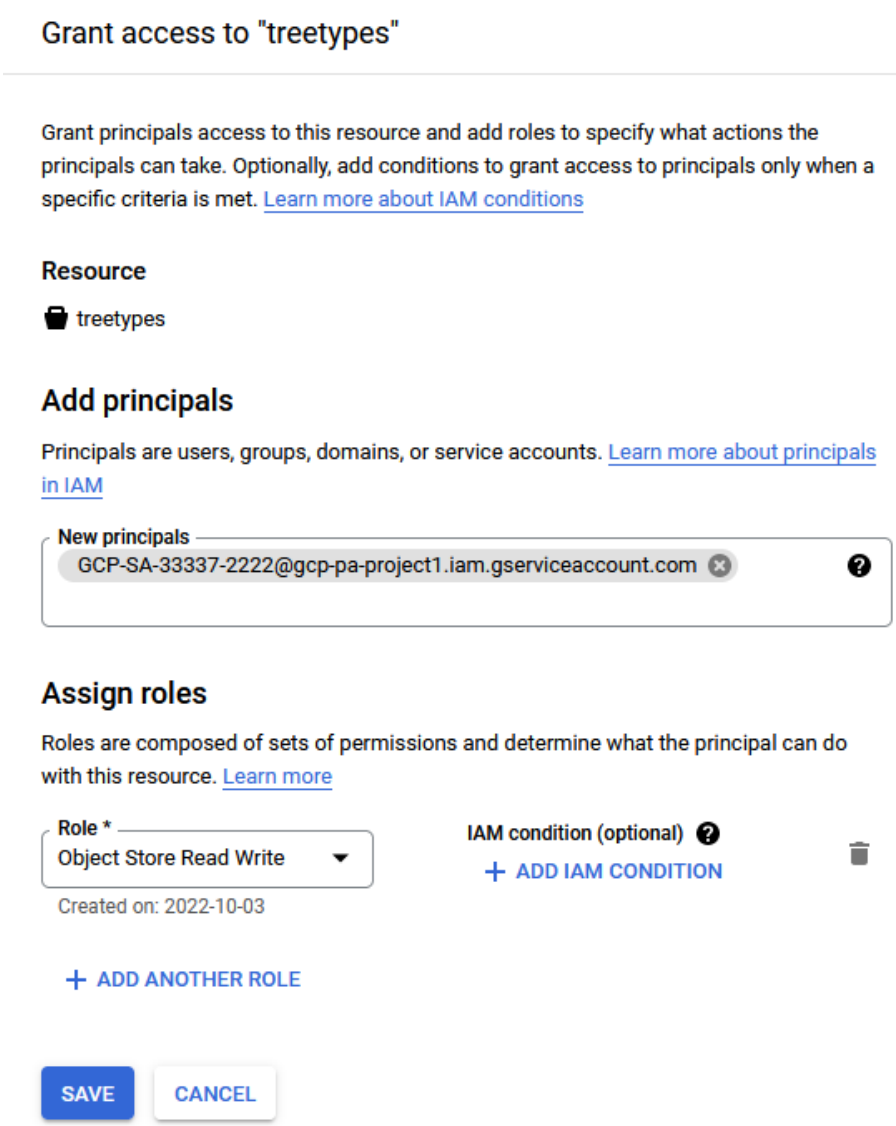
   b. Select the bucket you want to use, and click **PERMISSIONS**.

   c. Click **+ ADD PRINCIPAL**.

6. On the Grant access to "*bucketname*" dialog, add roles and a principals for the selected resource.

   a. Under **Add principals** add the value of the `gcp_service_account` parameter from your Autonomous Database instance.

b. On the Grant access to "*bucketname*" dialog, enter roles under **Assign Roles** and then click **SAVE**.



After you complete these steps the roles and principals are assigned. This allows your application running on the Autonomous Database instance to access the GCP resource with a Google service account.

Use Google Service Account with DBMS_CLOUD

When you make `DBMS_CLOUD` calls to access Google Cloud Platform (GCP) resources and specify the credential name as `GCP$PA`, the authentication on the Google Cloud Platform side happens using a Google service account.

If you have not already done so, perform the prerequisite steps:

- Enable the ADMIN schema or another schema to use Google service account authentication. See Enable Google Service Account and Find the GCP Service Account Name for more information.

- Perform the Google Cloud Platform role assignments for the resources you want to access. See Assign Roles to the Google Service Account and Provide Access for GCP Resources for more information.

To use a `DBMS_CLOUD` procedure or function with Google service account authentication:

1. Use `GCP$PA` as the credential name.
2. Construct the URI to access the GCP resource using virtual hosted-style:

   `https://BUCKET_NAME.storage.googleapis.com/OBJECT_NAME`

   For example, you can access Google Cloud Storage using Google service account credentials as follows:

```
SELECT * FROM DBMS_CLOUD.LIST_OBJECTS('GCP$PA', 'https://
treetypes.storage.googleapis.com/' );

OBJECT_NAME BYTES CHECKSUM                          CREATED LAST_MODIFIED
----------- ----- -------------------------------- -------
------------------------
trees.txt      58 682075a8c38f5686c32c25c6fb67dcbe
2022-10-05T20:03:55.253Z
```

See the following for more information:

- See Request endpoints for more information on GCP virtual hosted-style requests.
- See LIST_OBJECTS Function.

Disable Google Service Account

To disable Google service account access to Google Cloud Platform (GCP) resources, use `DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH`.

When the `provider` value is `GCP` and the `username` is a user other than the `ADMIN` user, the procedure revokes the privileges from the specified user. In this case, the `ADMIN` user and other users can continue to use `GCP$PA`.

For example, to revoke privileges for `adb_user`:

```
BEGIN
    DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH(
        provider => 'GCP',
        username => 'adb_user');
END;
/
```

When the `provider` value is `GCP` and the `username` is `ADMIN`, the procedure disables Google service account access on the Autonomous Database instance. The default value for `username` is `ADMIN`.

For example:

```
BEGIN
    DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH(
        provider => 'GCP' );
```

```
END;
/
```

See DISABLE_PRINCIPAL_AUTH Procedure for more information.

Google Service Account Notes

Notes for using Google service account.

- **Google Cloud Platform (GCP) character restriction**: `DBMS_CLOUD` does not support a URI containing an "_" to access a Google Cloud Storage bucket name. If your Google Cloud Storage bucket name contains an "_", you might see the following error:

  ```
  SELECT * FROM DBMS_CLOUD.LIST_OBJECTS('GCP$PA', 'https://
  app_bucket.storage.googleapis.com/');

  ORA-20006: Unsupported object store URI - https://
  app_bucket.storage.googleapis.com/
  ORA-06512: at "C##CLOUD$SERVICE.DBMS_CLOUD", line 1306
  ```

- **Cloning an Autonomous Database instance with a Google service account**: When you clone an instance with a Google service account enabled, the Google service account configuration is not carried over to the clone. Perform the steps to enable the Google service account on the clone if you want to enable Google service account on a cloned instance.

Load Data into Autonomous Database

Describes packages and tools to load data with Autonomous Database.

- About Data Loading
- Load Data with Oracle Database Actions
  Oracle Database Actions provides a web-based interface with development tools, data tool, administration, and monitoring features and lets you load or access data from local files, from cloud storage, or from remote databases.
- Load Data from Files in the Cloud
  The PL/SQL package `DBMS_CLOUD` provides support for loading data from text, ORC, Parquet, and Avro files in the Cloud to your tables in Autonomous Database. In addition, using `DBMS_CLOUD` you can load Data Pump dump files in the Cloud to your tables in Autonomous Database.
- Import Data Using Oracle Data Pump on Autonomous Database
  Oracle Data Pump offers very fast bulk data and metadata movement between Oracle databases and Autonomous Databases.
- Load Data from Local Files with Oracle Database Actions
  Using Oracle Database Actions, from the Worksheet page, you can load data from local files into an existing table.
- Load Data or Query Data from Files in a Directory
  You can use `DBMS_CLOUD` procedures to load data from files in a directory, including directories created on attached network file systems. You can also use these procedures to create external tables that you can use to query data.

- Load Data from Local Files Using SQL*Loader
  Instead of using SQL*Loader Oracle recommends loading data from the Cloud Object Storage for better performance and enhanced functionality.
- Using Data Pipelines for Continuous Load and Export
  Data pipelines allow you to repeatedly load data from object store or export data to object store. Load pipelines provide continuous incremental data loading from external sources (as data arrives on object store it is loaded to a database table). Export pipelines provide continuous incremental data exporting to object store (as new data appears in a database table it is exported to object store).
- The Data Load Page
  Use the Data Load page to make more data available to your Oracle Autonomous Database. You can load data from files or databases, from links to external databases or cloud storage files, or from a live feed of data from cloud storage.
- Transform Data Using Data Studio
  Data Transforms is an easy-to-use graphical user interface that you can use to design graphical data transformations for data integration with Autonomous Database.
- View Data Pump Jobs and Import Data with Data Pump
  Use the Data Pump page to view Data Pump jobs and use the wizard to quickly create and run Data Pump import jobs.

About Data Loading

Autonomous Database provides many different options to load data into your database.

You can load data using any of the following methods:

- You can load data using Oracle Database Actions.
- You can load data using Oracle Database tools and Oracle or other 3rd party data integration tools.
- On transaction processing systems you traditionally ingest data through routine transactions or with DML operations.
- You can load data using `DBMS_CLOUD` procedures.

In general you load data from files local to your client computer or from files stored in a cloud-based object store, or in connected file systems. To load data from files in the cloud, use either Oracle Database Actions or use the Autonomous Database PL/SQL package `DBMS_CLOUD` to load files from the cloud, in connected file systems, or the local file system. `DBMS_CLOUD` provides parallel execution procedures for bulk file upload, download, copy, and transfer activities, which streamlines the user experience and delivers optimal performance for bulk file operations.

For the fastest data loading experience Oracle recommends uploading the source files to a cloud-based object store, such as Oracle Cloud Infrastructure Object Storage, before loading the data into your database. Oracle provides support for loading files that are located locally in your data center, but when using this method of data loading you should factor in the transmission speeds across the Internet which may be significantly slower.

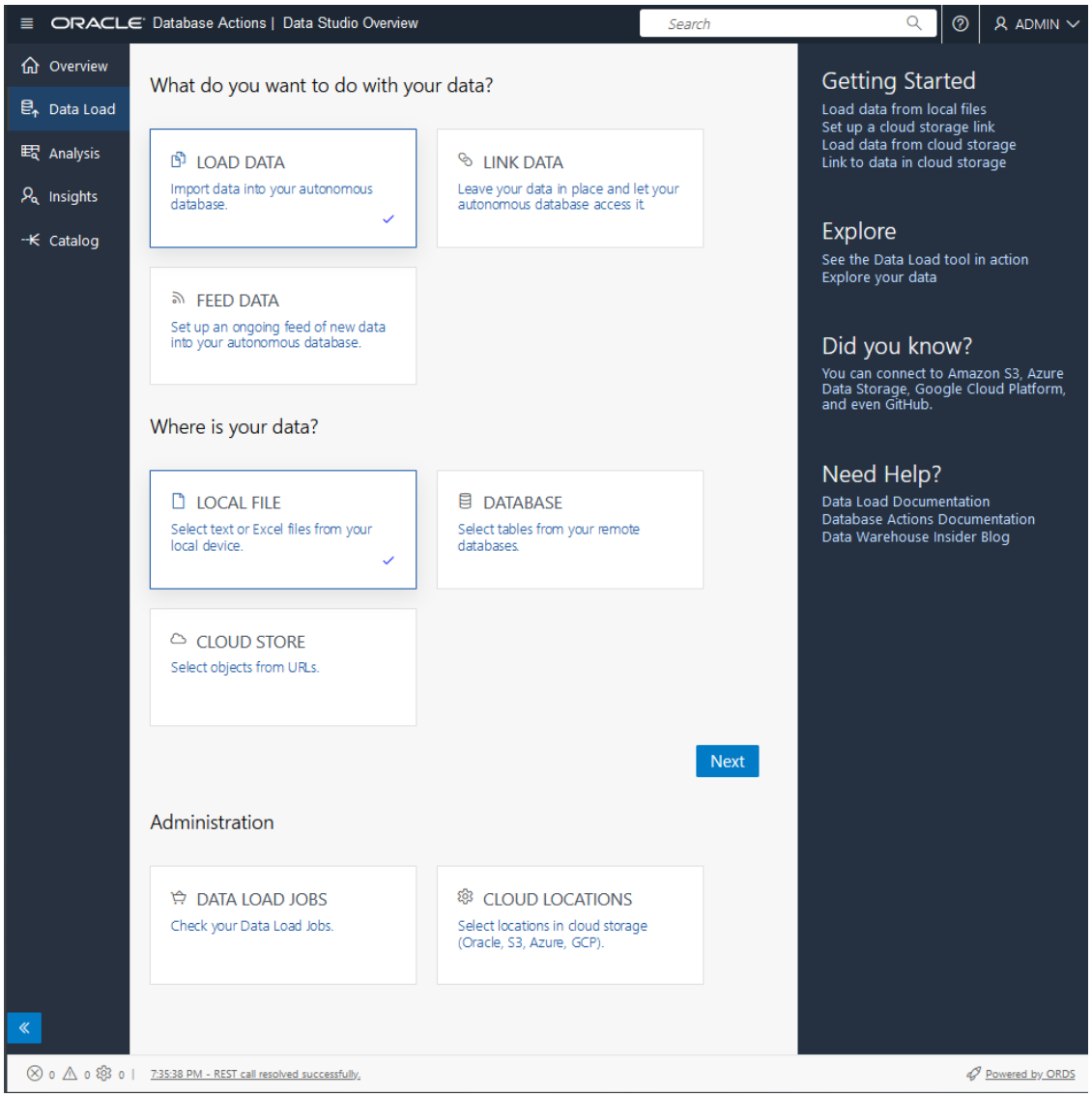
For more information on Oracle Cloud Infrastructure Object Storage, see Putting Data into Object Storage and Overview of Object Storage.

> **Note:**
>
> If you are not using `ADMIN` user, ensure the user has the necessary privileges for the operations the user needs to perform. See Manage User Privileges on Autonomous Database - Connecting with a Client Tool for more information.

Load Data with Oracle Database Actions

Oracle Database Actions provides a web-based interface with development tools, data tool, administration, and monitoring features and lets you load or access data from local files, from cloud storage, or from remote databases.

On the Database Actions Data Load page you can choose to load data from a file on your local device, from cloud storage, or from a database. You can also choose to explore the data in your Oracle Autonomous Database. See The Data Load Page for detailed information and the steps for loading data using Database Actions.



See Connect with Built-In Oracle Database Actions for information on accessing Oracle Database Actions.

Load Data from Files in the Cloud

The PL/SQL package `DBMS_CLOUD` provides support for loading data from text, ORC, Parquet, and Avro files in the Cloud to your tables in Autonomous Database. In addition, using `DBMS_CLOUD` you can load Data Pump dump files in the Cloud to your tables in Autonomous Database.

The package `DBMS_CLOUD` supports loading files from the following cloud services:

- Oracle Cloud Infrastructure Object Storage
- Azure Blob Storage
- Amazon S3
- Amazon S3-Compatible, including: Wasabi Hot Cloud Storage
- GitHub Repository
- Google Cloud Storage

See DBMS_CLOUD Package File URI Formats for more information.

- Create Credentials and Copy Data into an Existing Table
  For data loading from files in the Cloud, you need to first store your object storage credentials in your Autonomous Database and then use the procedure `DBMS_CLOUD.COPY_DATA` to load data.
- Create Credentials and Load Data Pump Dump Files into an Existing Table
  For data loading you can also use Oracle Data Pump dump files as source files.
- Load JSON on Autonomous Database
  The PL/SQL procedure `DBMS_CLOUD.COPY_COLLECTION` provides support for loading JSON documents into SODA collections. The procedure `DBMS_CLOUD.COPY_DATA` provides support for loading JSON data into an existing table in Autonomous Database.
- Monitor and Troubleshoot Loads
  All data load operations done using the PL/SQL package `DBMS_CLOUD` are logged in the tables `dba_load_operations` and `user_load_operations`:
- Examples for Loading Data into Autonomous Database
  Provides examples for loading data in various formats and from different Cloud Storage providers into Autonomous Database.

Create Credentials and Copy Data into an Existing Table

For data loading from files in the Cloud, you need to first store your object storage credentials in your Autonomous Database and then use the procedure `DBMS_CLOUD.COPY_DATA` to load data.

The source file in this example, `channels.txt`, has the following data:

```
S,Direct Sales,Direct
T,Tele Sales,Direct
C,Catalog,Indirect
I,Internet,Indirect
P,Partners,Others
```

1. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`. For example:

```
SET DEFINE OFF
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adb_user@example.com',
    password => 'password'
  );
END;
/
```

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for all data loads.

For detailed information about the parameters, see CREATE_CREDENTIAL Procedure.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

> **Note:**
>
> Some tools like SQL*Plus and SQL Developer use the ampersand character (`&`) as a special character. If you have the ampersand character in your password use the `SET DEFINE OFF` command in those tools as shown in the example to disable the special character and get the credential created properly.

2. Load data into an existing table using the procedure `DBMS_CLOUD.COPY_DATA`. For example:

```
CREATE TABLE CHANNELS
   (channel_id CHAR(1),
    channel_desc VARCHAR2(20),
    channel_class VARCHAR2(20)
   );
/

BEGIN
 DBMS_CLOUD.COPY_DATA(
    table_name =>'CHANNELS',
    credential_name =>'DEF_CRED_NAME',
    file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/channels.txt',
    format => json_object('delimiter' value ',')
 );
END;
/
```

The parameters are:

- `table_name`: is the target table's name.

- `credential_name`: is the name of the credential created in the previous step. The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
- `file_uri_list`: is a comma delimited list of the source files you want to load.
- `format`: defines the options you can specify to describe the format of the source file, including whether the file is of type text, ORC, Parquet, or Avro.

  If the data in your source files is encrypted, decrypt the data by specifying the `format` parameter with the `encryption` option. See Decrypt Data While Importing from Object Storage for more information on decrypting data.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

> **Note:**
>
> Autonomous Database supports a variety of source file formats, including compressed data formats. See DBMS_CLOUD Package Format Options and the `DBMS_CLOUD compression` format option to see the supported compression types.

For detailed information about the parameters, see COPY_DATA Procedure and COPY_DATA Procedure for Avro, ORC, or Parquet Files.

Create Credentials and Load Data Pump Dump Files into an Existing Table

For data loading you can also use Oracle Data Pump dump files as source files.

The source files for this load type must be exported from the source system using the `ORACLE_DATAPUMP` access driver in External Tables. See Unloading and Loading Data with the ORACLE_DATAPUMP Access Driver for details on exporting using the `ORACLE_DATAPUMP` access driver.

To load the data you first move the dump files that were exported using the `ORACLE_DATAPUMP` access driver to your Object Store and then use `DBMS_CLOUD.COPY_DATA` to load the dump files to an existing table in your database.

The source files in this example are the Oracle Data Pump dump files, `exp01.dmp` and `exp02.dmp`.

1. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`. For example:

```
SET DEFINE OFF
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adb_user@example.com',
    password => 'password'
  );
END;
/
```

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for all data loads.

For detailed information about the parameters, see CREATE_CREDENTIAL Procedure.

> **Note:**
>
> Some tools like SQL*Plus and SQL Developer use the ampersand character (`&`) as a special character. If you have the ampersand character in your password use the `SET DEFINE OFF` command in those tools as shown in the example to disable the special character and get the credential created properly.

2. Load data into an existing table using the procedure `DBMS_CLOUD.COPY_DATA`. For example:

```
CREATE TABLE CHANNELS
   (channel_id CHAR(1),
    channel_desc VARCHAR2(20),
    channel_class VARCHAR2(20)
   );
/

BEGIN
 DBMS_CLOUD.COPY_DATA(
    table_name =>'CHANNELS',
    credential_name =>'DEF_CRED_NAME',
    file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/exp01.dmp,
                     https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/exp02.dmp',
    format => json_object('type' value 'datapump')
 );
END;
/
```

The parameters are:

- `table_name`: is the target table's name.
- `credential_name`: is the name of the credential created in the previous step. The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
- `file_uri_list`: is a comma delimited list of the Data Pump dump files you want to load.
- `format`: defines the options you can specify to describe the format of the source file. When you specify the `type` as `'datapump'`, the only other valid format parameter is `'rejectlimit'`.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

For detailed information about the parameters, see COPY_DATA Procedure and COPY_DATA Procedure for Avro, ORC, or Parquet Files.

Load JSON on Autonomous Database

The PL/SQL procedure `DBMS_CLOUD.COPY_COLLECTION` provides support for loading JSON documents into SODA collections. The procedure `DBMS_CLOUD.COPY_DATA` provides support for loading JSON data into an existing table in Autonomous Database.

- About Loading JSON Documents
  You load SODA collections into Autonomous Database using the PL/SQL procedure `DBMS_CLOUD.COPY_COLLECTION` and you load JSON data into a table using `DBMS_CLOUD.COPY_DATA`.
- Load a JSON File of Line-Delimited Documents into a Collection
  For loading data from collections in the Cloud, you must first store your object storage credentials in your Autonomous Database and then use the procedure `DBMS_CLOUD.COPY_COLLECTION` to load documents into a collection.
- Load an Array of JSON Documents into a Collection
  To load data from collections in the Cloud, you first store your object storage credentials in your Autonomous Database and then use PL/SQL procedure `DBMS_CLOUD.COPY_COLLECTION` to load documents into a collection. This topic explains how to load documents to your database from a JSON array in a file.
- Monitor and Troubleshoot COPY_COLLECTION Loads
  All data load operations you perform using the PL/SQL package `DBMS_CLOUD` are logged in the tables `dba_load_operations` and `user_load_operations`. Use these tables to monitor loading with `DBMS_CLOUD.COPY_COLLECTION`.
- Textual JSON Objects That Represent Extended Scalar Values
  Native binary JSON data (OSON format) extends the JSON language by adding scalar types, such as date, that correspond to SQL types and are not part of the JSON standard. Oracle Database also supports the use of textual JSON *objects* that *represent* JSON scalar values, including such nonstandard values.
- Create Credentials and Copy JSON Data into an Existing Table
  Use `DBMS_CLOUD.COPY_DATA` to load JSON data in the cloud into a table.

About Loading JSON Documents

You load SODA collections into Autonomous Database using the PL/SQL procedure `DBMS_CLOUD.COPY_COLLECTION` and you load JSON data into a table using `DBMS_CLOUD.COPY_DATA`.

- `DBMS_CLOUD.COPY_COLLECTION` supports the following typical document loading procedures:
  - Loading line-delimited JSON into a collection. See Load a JSON File of Line-Delimited Documents into a Collection for this procedure.
  - Loading an array of JSON documents into a collection. See Load an Array of JSON Documents into a Collection for this procedure.
- `DBMS_CLOUD.COPY_DATA` supports the following for loading from JSON data in Object Store:
  - Create Credentials and Copy JSON Data into an Existing Table

Load a JSON File of Line-Delimited Documents into a Collection

For loading data from collections in the Cloud, you must first store your object storage credentials in your Autonomous Database and then use the procedure `DBMS_CLOUD.COPY_COLLECTION` to load documents into a collection.

This example loads JSON values from a line-delimited file and uses the JSON file `myCollection.json`. Each value, each line, is loaded into a collection on your database as a single document.

Here's an example of such a file. It has three lines, with one object per line. Each of those objects gets loaded as a separate JSON document.

```
{ "name" : "apple", "count": 20 }
{ "name" : "orange", "count": 42 }
{ "name" : "pear", "count": 10 }
```

Before loading the data from `myCollection.json` into your database, copy the file to your object store:

- Create a bucket in the object store. For example, create an Oracle Cloud Infrastructure Object Storage bucket from the Oracle Cloud Infrastructure Object Storage link, and then in your selected compartment click **Create Bucket**, or use a command such as the following OCI CLI command to create a bucket:

  ```
  oci os bucket create --name fruit_bucket -c <compartment id>
  ```

- Copy the JSON file to your object store bucket. For example use the following OCI CLI command to copy the JSON file to the `fruit_bucket` on Oracle Cloud Infrastructure Object Storage:

  ```
  oci os object put --bucket-name fruit_bucket \
                    --file "myCollection.json"
  ```

Load the JSON file from object store into a collection named `fruit` on your database as follows:

1. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`, as shown in the following example:

   ```
   SET DEFINE OFF
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DEF_CRED_NAME',
       username => 'adb_user@example.com',
       password => 'password'
     );
   END;
   /
   ```

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your

object store credentials change. Once you store the credentials, you can use the same credential name for loading all documents.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

See CREATE_CREDENTIAL Procedure for detailed information about the parameters.

> **Note:**
>
> Some tools like SQL*Plus and SQL Developer use the ampersand character (`&`) as a special character. If you have the ampersand character in your password, then use the `SET DEFINE OFF` command in those tools as shown in the example to disable the special character, and get the credential created properly.

2. Load the data into a collection using the procedure `DBMS_CLOUD.COPY_COLLECTION`.

```
BEGIN
 DBMS_CLOUD.COPY_COLLECTION(
    collection_name => 'fruit',
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   =>
      'https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/
fruit_bucket/o/myCollection.json',
    format          =>
      JSON_OBJECT('recorddelimiter' value '''\n''')  );
END;
/
```

The parameters are:

- `collection_name`: is the name of the target collection.
- `credential_name`: is the name of the credential created in the previous step. The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
- `file_uri_list`: is a comma delimited list of the source files that you want to load.
- `format`: defines the options that you can specify to describe the format of the source file. The format options `characterset`, `compression`, `encryption`, `ignoreblanklines`, `jsonpath`, `maxdocsize`, `recorddelimiter`, `rejectlimit`, `type`, `unpackarrays` are supported while loading JSON data. Any other formats specified will result in an error.

  If the data in your source files is encrypted, decrypt the data by specifying the `format` parameter with the `encryption` option. See Decrypt Data While Importing from Object Storage for more information on decrypting data.

  See DBMS_CLOUD Package Format Options for more information.

Where *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and `fruit_bucket` is the bucket name. See Understanding Object Storage Namespaces and Overview of Object Storage for more information.

For detailed information about the parameters, see COPY_COLLECTION Procedure.

The collection `fruit` on your database now contains one document for each line in the file `myCollection.json`.

Load an Array of JSON Documents into a Collection

To load data from collections in the Cloud, you first store your object storage credentials in your Autonomous Database and then use PL/SQL procedure `DBMS_CLOUD.COPY_COLLECTION` to load documents into a collection. This topic explains how to load documents to your database from a JSON array in a file.

> **Note:**
>
> You can also load documents from a JSON array in a file into a collection using SODA for REST. See Load Purchase-Order Sample Data Using SODA for REST.

This example uses the JSON file `fruit_array.json`. The following shows the contents of the file `fruit_array.json`:

```
[{"name" : "apple", "count": 20 },
 {"name" : "orange", "count": 42 },
 {"name" : "pear", "count": 10 }]
```

Before loading data into Autonomous Database, copy the data to your object store as follows:

- Create a bucket in the object store. For example, create an Oracle Cloud Infrastructure Object Store bucket from the Oracle Cloud Infrastructure Object Storage link, in your selected Compartment, by clicking **Create Bucket**, or use a command line tool such as the following OCI CLI command:

  ```
  oci os bucket create -name json_bucket -c <compartment id>
  ```

- Copy the JSON file to the object store. For example, the following OCI CLI command copies the JSON file `fruit_array.json` to the object store:

  ```
  oci os object put --bucket-name json_bucket --file "fruit_array.json"
  ```

Load the JSON file from object store into a SODA collection named `fruit2` on your database:

1. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`, as shown in the following example:

   ```
   SET DEFINE OFF
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DEF_CRED_NAME',
       username => 'adb_user@example.com',
       password => 'password'
     );
   END;
   /
   ```

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials, you can use the same credential name for loading all documents.

See CREATE_CREDENTIAL Procedure for detailed information about the parameters.

> **Note:**
>
> Some tools like SQL*Plus and SQL Developer use the ampersand character (`&`) as a special character. If you have the ampersand character in your password, then use the `SET DEFINE OFF` command in those tools as shown in the example to disable the special character, and get the credential created properly.

2. Load the data into a collection using the procedure `DBMS_CLOUD.COPY_COLLECTION`.

```
BEGIN
  DBMS_CLOUD.COPY_COLLECTION(
    collection_name => 'fruit2',
    credential_name => 'DEF_CRED_NAME',
    file_uri_list => 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/
namespace-string/b/json/o/fruit_array.json',
    format => '{"recorddelimiter" : "0x''01''", "unpackarrays" : "TRUE",
"maxdocsize" : "10240000"}'
  );
END;
/
```

In this example you load a single JSON value which occupies the whole file, so there is no need to specify a record delimiter. To indicate that there is no record delimiter, you can use a character that does not occur in the input file. For this example, to indicate that there is no delimiter, the control character 0x01 (SOH) is set to load the JSON documents into a collection,. Thus, you specify a value for the `recorddelimiter` that does not occur in the JSON file. For example, you can use value `"0x''01''"` because this character does not occur directly in JSON text.

When `unpackarrays` parameter for format value is set to `TRUE`, the array of documents is loaded as individual documents rather than as an entire array. The unpacking of array elements is however limited to single level. If there are nested arrays in the documents, those arrays are not unpacked.

The parameters are:

- `collection_name`: is the name of the target collection.
- `credential_name`: is the name of the credential created in the previous step. The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
- `file_uri_list`: is a comma delimited list of the source files that you want to load.
- `format`: defines the options that you can specify to describe the format of the source file. The format options `characterset`, `compression`, `encryption`, `ignoreblanklines`, `jsonpath`, `maxdocsize`, `recorddelimiter`, `rejectlimit`, `type`, `unpackarrays` are supported for loading JSON data. Any other formats specified will result in an error.

If the data in your source files is encrypted, decrypt the data by specifying the `format` parameter with the `encryption` option. See Decrypt Data While Importing from Object Storage for more information on decrypting data.

See DBMS_CLOUD Package Format Options for more information.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

For detailed information about the parameters, see COPY_COLLECTION Procedure.

The load of `fruit_array.json`, with `DBMS_CLOUD.COPY_COLLECTION` using the format option `unpackarrays` recognizes array values in the source and instead of loading the data as a single document, as it would by default, the data is loaded in the collection `fruit2` with each value in the array as a single document.

Monitor and Troubleshoot COPY_COLLECTION Loads

All data load operations you perform using the PL/SQL package `DBMS_CLOUD` are logged in the tables `dba_load_operations` and `user_load_operations`. Use these tables to monitor loading with `DBMS_CLOUD.COPY_COLLECTION`.

- `dba_load_operations` shows all load operations
- `user_load_operations` shows the load operations in your schema

You can query these tables to see information about ongoing and completed data loads. For example, the following `SELECT` statement with a `WHERE` clause predicate on the `TYPE` column shows load operations of the type `COPY`:

```
SELECT table_name, owner_name, type, status, start_time, update_time,
logfile_table, badfile_table
   FROM user_load_operations WHERE type = 'COPY';
TABLE_NAME  OWNER_NAME   TYPE    STATUS     START_TIME
UPDATE_TIME           LOGFILE_TABLE   BADFILE_TABLE
FRUIT       ADMIN        COPY    COMPLETED  2020-04-23 22:27:37    2020-04-23
22:27:38   ""             ""
FRUIT       ADMIN        COPY    FAILED     2020-04-23 22:28:36    2020-04-23
22:28:37   COPY$2_LOG      COPY$2_BAD
```

The `LOGFILE_TABLE` column shows the name of the table you can query to look at the log of a load operation. For example, the following query shows the log of the load operation with status `FAILED` and timestamp `2020-04-23 22:28:36`:

```
SELECT * FROM COPY$2_LOG;
```

The column `BADFILE_TABLE` shows the name of the table you can query to review information for the rows reporting errors during loading. For example, the following query shows the rejected records for the load operation:

```
SELECT * FROM COPY$2_BAD;
```

Depending on the errors shown in the log and the rows shown in the `BADFILE_TABLE` table, you might be able to correct errors by specifying different format options with `DBMS_CLOUD.COPY_COLLECTION`.

> **Note:**
>
> The `LOGFILE_TABLE` and `BADFILE_TABLE` tables are stored for two days for each load operation and then removed automatically.

See DELETE_ALL_OPERATIONS Procedure for information on clearing the `user_load_operations` table.

Textual JSON Objects That Represent Extended Scalar Values

Native binary JSON data (OSON format) extends the JSON language by adding scalar types, such as date, that correspond to SQL types and are not part of the JSON standard. Oracle Database also supports the use of textual JSON *objects* that *represent* JSON scalar values, including such nonstandard values.

When you create native binary JSON data from textual JSON data that contains such **extended objects**, they can optionally be *replaced* with corresponding (native binary) JSON scalar values.

An example of an extended object is `{"$numberDecimal":31}`. It represents a JSON scalar value of the nonstandard type *decimal number*, and when interpreted as such it is replaced by a decimal number in native binary format.

For example, when you use the JSON data type constructor, `JSON`, if you use keyword `EXTENDED` then recognized extended objects in the textual input are replaced with corresponding scalar values in the native binary JSON result. If you do not include keyword `EXTENDED` then no such replacement occurs; the textual extended JSON objects are simply converted as-is to JSON objects in the native binary format.

In the opposite direction, when you use Oracle SQL function `json_serialize` to serialize binary JSON data as textual JSON data (`VARCHAR2`, `CLOB`, or `BLOB`), you can use keyword `EXTENDED` to replace (native binary) JSON scalar values with corresponding textual extended JSON objects.

> **Note:**
>
> If the database you use is an Oracle Autonomous Database then you can use PL/SQL procedure `DBMS_CLOUD.copy_collection` to create a JSON document collection from a file of JSON data such as that produced by common NoSQL databases, including Oracle NoSQL Database.
>
> If you use **`ejson`** as the value of the `type` parameter of the procedure, then recognized extended JSON objects in the input file are replaced with corresponding scalar values in the resulting native binary JSON collection. In the other direction, you can use function `json_serialize` with keyword `EXTENDED` to replace scalar values with extended JSON objects in the resulting textual JSON data.

These are the two main use cases for extended objects:

- *Exchange* (import/export):
  - Ingest existing JSON data (from somewhere) that contains extended objects.

  - Serialize native binary JSON data as textual JSON data with extended objects, for some use outside the database.

- *Inspection* of native binary JSON data: see what you have by looking at corresponding extended objects.

For exchange purposes, you can ingest JSON data from a file produced by common NoSQL databases, including Oracle NoSQL Database, converting extended objects to native binary JSON scalars. In the other direction, you can export native binary JSON data as textual data, replacing Oracle-specific scalar JSON values with corresponding textual extended JSON objects.

As an example of inspection, consider an object such as `{"dob" : "2000-01-02T00:00:00"}` as the result of serializing native JSON data. Is `"2000-01-02T00:00:00"` the result of serializing a native binary value of type date, or is the native binary value just a string? Using `json_serialize` with keyword `EXTENDED` lets you know.

The mapping of extended object fields to scalar JSON types is, in general, many-to-one: more than one kind of extended JSON object can be mapped to a given scalar value. For example, the extended JSON objects `{"$numberDecimal":"31"}` and `{"$numberLong:"31"}` are both translated as the value 31 of JSON-language scalar type number, and item method `type()` returns `"number"` for each of those JSON scalars.

Item method `type()` reports the JSON-language scalar type of its targeted value (as a JSON string). Some scalar values are distinguishable internally, even when they have the same scalar type. This generally allows function `json_serialize` (with keyword `EXTENDED`) to reconstruct the original extended JSON object. Such scalar values are distinguished internally either by using *different SQL types* to implement them or by *tagging them with the kind of extended JSON object* from which they were derived.

When `json_serialize` reconstructs the original extended JSON object the result is not always *textually* identical to the original, but it is always *semantically* equivalent. For example, `{"$numberDecimal":"31"}` and `{"$numberDecimal":31}` are semantically equivalent, even though the field values differ in type (string and number). They are translated to the same internal value, and each is tagged as being derived from a `$numberDecimal` extended object (same tag). But when serialized, the *result for both* is `{"$numberDecimal":31}`. Oracle always uses the most directly relevant type for the field value, which in this case is the JSON-language value `31`, of scalar type number.

> **Note:**
>
> There are two cases where the type of the original extended object can be *lost* when deriving the internal binary-JSON value.
>
> - An extended object with field `$numberInt` is translated to an Oracle SQL `NUMBER` internal value, with no tag. Serializing that value produces a standard JSON-language value of type number. There is no loss in the numerical value; the only loss is the information that the original textual data was a `$numberInt` extended object.
> - Use of field `$numberDecimal` with infinite, very small, very large, or not-a-number values is *unsupported*, and results in undefined behavior. *Do not use* a string value that represents positive infinity (`"Infinity"` or `"Inf"`), negative infinity (`"-Infinity"` or `"-Inf"`), or an unknown value (not a number, `"Nan"`) with `$numberDecimal` — instead, use `$numberDouble` with such values.

Table 3-1 presents correspondences among the various types used. It maps across (1) types of extended objects used as input, (2) types reported by item method `type()`, (3) SQL types used internally, (4) standard JSON-language types used as output by function `json_serialize`, and (5) types of extended objects output by `json_serialize` when keyword `EXTENDED` is specified.

Table 3-1 Extended JSON Object Type Relations

Extended Object Type (Input)	Oracle JSON Scalar Type (Reported by type())	SQL Scalar Type	Standard JSON Scalar Type (Output)	Extended Object Type (Output)
`$numberDouble` with value a JSON number, a string representing the number, or one of these strings: `"Infinity"`, `"-Infinity"`, `"Inf"`, `"-Inf"`, `"Nan"`[1]	double	`BINARY_DOUBLE`	number	`$numberDouble` with value a JSON number or one of these strings: `"Inf"`, `"-Inf"`, `"Nan"`[2]
`$numberFloat` with value the same as for `$numberDouble`	float	`BINARY_FLOAT`	number	`$numberFloat` with value the same as for `$numberDouble`
`$numberDecimal` with value the same as for `$numberDouble`	number	`NUMBER`	number	`$numberDecimal` with value the same as for `$numberDouble`
`$numberInt` with value a signed 32-bit integer or a string representing the number	number	`NUMBER`	number	`$numberInt` with value the same as for `$numberDouble`
`$numberLong` with value a JSON number or a string representing the number	number	`NUMBER`	number	`$numberLong` with value the same as for `$numberDouble`
`$binary` with value one of these: • a string of base-64 characters • An object with fields `base64` and `subType`, whose values are a string of base-64 characters and the number `0` (arbitrary binary) or `4` (UUID), respectively When the value is a string of base-64 characters, the extended object can also have field `$subtype` with value 0 or 4, expressed as a one-byte integer (0-255) or a 2-character hexadecimal string. representing such an integer	binary	`BLOB` or `RAW`	string Conversion is equivalent to the use of SQL function `rawtohex`.	One of the following: • `$binary` with value a string of base-64 characters • `$rawid` with value a string of 32 hexadecimal characters, if input had a `subType` value of `4` (UUID)
`$oid` with value a string of 24 hexadecimal characters	binary	`RAW(12)`	string Conversion is equivalent to the use of SQL function `rawtohex`.	`$rawid` with value a string of 24 hexadecimal characters
`$rawhex` with value a string with an even number of hexadecimal characters	binary	`RAW`	string Conversion is equivalent to the use of SQL function `rawtohex`.	`$binary` with value a string of base-64 characters, right-padded with = characters

Table 3-1 (Cont.) Extended JSON Object Type Relations

Extended Object Type (Input)	Oracle JSON Scalar Type (Reported by type())	SQL Scalar Type	Standard JSON Scalar Type (Output)	Extended Object Type (Output)
`$rawid` with value a string of 24 or 32 hexadecimal characters	binary	`RAW`	string Conversion is equivalent to the use of SQL function `rawtohex`.	`$rawid`
`$oracleDate` with value an ISO 8601 date string	date	`DATE`	string	`$oracleDate` with value an ISO 8601 date string
`$oracleTimestamp` with value an ISO 8601 timestamp string	timestamp	`TIMESTAMP`	string	`$oracleTimestamp` with value an ISO 8601 timestamp string
`$oracleTimestampTZ` with value an ISO 8601 timestamp string with a numeric time zone offset or with `Z`	timestamp with time zone	`TIMESTAMP WITH TIME ZONE`	string	`$oracleTimestampTZ` with value an ISO 8601 timestamp string with a numeric time zone offset or with `Z`
`$date` with value one of the following: • An integer millisecond count since January 1, 1990 • An ISO 8601 timestamp string • An object with field `numberLong` with value an integer millisecond count since January 1, 1990	timestamp with time zone	`TIMESTAMP WITH TIME ZONE`	string	`$oracleTimestampTZ` with value an ISO 8601 timestamp string with a numeric time zone offset or with `Z`
`$intervalDaySecond` with value an ISO 8601 interval string as specified for SQL function `to_dsinterval`	daysecondInterval	`INTERVAL DAY TO SECOND`	string	`$intervalDaySecond` with value an ISO 8601 interval string as specified for SQL function `to_dsinterval`
`$intervalYearMonth` with value an ISO 8601 interval string as specified for SQL function `to_yminterval`	yearmonthInterval	`INTERVAL YEAR TO MONTH`	string	`$intervalYearMonth` with value an ISO 8601 interval string as specified for SQL function `to_yminterval`

[1] The string values are interpreted case-insensitively. For example, `"NAN"` `"nan"`, and `"nAn"` are accepted and equivalent, and similarly `"INF"`, `"inFinity"`, and `"iNf"`. Infinitely large (`"Infinity"` or `"Inf"`) and small (`"-Infinity"` or `"-Inf"`) numbers are accepted with either the full word or the abbreviation.

[2] On output, only these string values are used — no full-word *Infinity* or letter-case variants.

> **See Also:**
>
> IEEE Standard for Floating-Point Arithmetic (IEEE 754)

Create Credentials and Copy JSON Data into an Existing Table

Use `DBMS_CLOUD.COPY_DATA` to load JSON data in the cloud into a table.

The source file in this example is a JSON data file.

1. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`. For example:

```
SET DEFINE OFF
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adb_user@example.com',
    password => 'password'
  );
END;
/
```

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for all data loads.

   For detailed information about the parameters, see CREATE_CREDENTIAL Procedure.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

2. Load JSON data into an existing table using the procedure `DBMS_CLOUD.COPY_DATA`.

   For example:

```
CREATE TABLE WEATHER2
    (WEATHER_STATION_ID VARCHAR2(20),
     WEATHER_STATION_NAME VARCHAR2(50));
/

BEGIN
  DBMS_CLOUD.COPY_DATA(
      table_name      => 'WEATHER2',
      credential_name => 'DEF_CRED_NAME',
      file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/jsonfiles*',
      format          =>  JSON_OBJECT('type' value 'json', 'columnpath'
value '["$.WEATHER_STATION_ID",
          "$.WEATHER_STATION_NAME"]')
    );
END;
/
```

   The parameters are:

   - `table_name`: is the target table's name.
   - `credential_name`: is the name of the credential created in the previous step.
   - `file_uri_list`: is a comma delimited list of the source files you want to load. You can use wildcards in the file names in your URIs. The character "*" can be used as the wildcard for multiple characters, the character "?" can be used as the wildcard for a single character.

- `format`: for `DBMS_CLOUD.COPY_DATA` with JSON data, the `type` is `json`. Specify other format values to define the options to describe the format of the JSON source file. See DBMS_CLOUD Package Format Options for more information.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

For detailed information about the parameters, see COPY_DATA Procedure.

Monitor and Troubleshoot Loads

All data load operations done using the PL/SQL package `DBMS_CLOUD` are logged in the tables `dba_load_operations` and `user_load_operations`:

- `dba_load_operations`: shows all load operations.
- `user_load_operations`: shows the load operations in your schema.

Query these tables to see information about ongoing and completed data loads. Use a `SELECT` statement with a `WHERE` clause predicate on the `TYPE` column to show load operations with the type `COPY`.

For example:

```
SELECT table_name, owner_name, type, status, start_time, update_time, logfile_table,
badfile_table
   FROM user_load_operations WHERE type = 'COPY';

TABLE_NAME OWNER_NAME TYPE STATUS    START_TIME               UPDATE_TIME
LOGFILE_TABLE BADFILE_TABLE
---------- ---------- ---- --------- ------------------------ ------------------------
------------- -------------
TREEDATA   ADMIN      COPY COMPLETED 2022-10-20T23:15:19.990Z 2022-10-20T23:15:24.238Z
COPY$1_LOG    COPY$1_BAD
```

The `LOGFILE_TABLE` column shows the name of the table you can query to look at the log of a load operation. For example, the following query shows the log for this load operation:

```
SELECT * FROM COPY$1_LOG;
```

The column `BADFILE_TABLE` shows the name of the table you can query to look at the rows with errors during loading. For example, the following query shows the rejected records for the load operation. If there are not any rejected rows in the operation, the query does not show any rejected rows.

```
SELECT * FROM COPY$1_BAD;
```

Depending on the errors shown in the log and the rows shown in the `BADFILE_TABLE` file you can correct the error by specifying the correct format options in `DBMS_CLOUD.COPY_DATA`.

When the format `type` is "`datapump`", any rows rejected up to the specified `rejectlimit` are logged in the log file, but a `BADFILE_TABLE` is not generated.

By default the `LOGFILE_TABLE` and `BADFILE_TABLE` files are retained for two days and then automatically removed. You can change the number of retention days with the `logretention` option for the `format` parameter.

See DBMS_CLOUD Package Format Options for more information.

See DELETE_ALL_OPERATIONS Procedure for information on clearing the `user_load_operations` table.

Monitor and Troubleshoot Bulk File Operations

See Monitor and Troubleshoot Bulk File Loads for information on monitoring and troubleshooting for bulk file operations.

Monitor and Troubleshoot ORC, Parquet, or Avro File Loading

As with other data files, ORC, Parquet, and Avro data loads generate logs that are viewable in the tables `dba_load_operations` and `user_load_operations`. Each load operation adds a record to `dba`[*user*]`_load_operations` that indicates the table containing the logs.

The log table provides summary information about the load.

> **Note:**
>
> For ORC, Parquet, or Avro files, when the `format` parameter `type` is set to the value `orc`, `parquet` or `avro` the `BADFILE_TABLE` table is always empty.
>
> - `PRIMARY KEY` constraint errors throw an `ORA` error.
> - If data for a column encounters a conversion error, for example, the target column is not large enough to hold the converted value, the value for the column is set to `NULL`. This does not produce a rejected record.

Examples for Loading Data into Autonomous Database

Provides examples for loading data in various formats and from different Cloud Storage providers into Autonomous Database.

- Load Data into Autonomous Database from Oracle Cloud Infrastructure Object Storage
  This example shows you how to load data from Oracle Cloud Infrastructure Object Storage to Autonomous Database using SQL commands.
- Load Data into Autonomous Database from AWS S3
  This example shows you how to load data from Amazon S3 object storage to Autonomous Database using SQL commands.
- Load Data into Autonomous Database from Azure Blob Storage
  This example shows you how to load data from Azure Blob Storage to Autonomous Database using SQL commands.
- Load a Fixed-Width File into a New Table
  This provides an example using `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` to load data from a fixed-width source file to an external table.

Load Data into Autonomous Database from Oracle Cloud Infrastructure Object Storage

This example shows you how to load data from Oracle Cloud Infrastructure Object Storage to Autonomous Database using SQL commands.

This example uses Oracle Cloud Infrastructure access credentials for user account authentication and an object URL for accessing the object in your Oracle Cloud Infrastructure Object Storage bucket.

To load data from Oracle Cloud Infrastructure Object Storage:

1. Create credentials for Oracle Cloud Infrastructure user account in the Autonomous Database.
2. Copy data from Oracle Cloud Infrastructure Object Storage to the database.

Topics

- Prepare for Loading Data from Oracle Cloud Infrastructure
  Verify the prerequisites and prepare for loading data from Oracle Cloud Infrastructure Object Storage.
- Steps for Loading Data from Oracle Cloud Infrastructure
  Run these steps to load data from Oracle Cloud Infrastructure Object Storage to Autonomous Database.

Prepare for Loading Data from Oracle Cloud Infrastructure

Verify the prerequisites and prepare for loading data from Oracle Cloud Infrastructure Object Storage.

Prerequisites

A data file, for example, `oci-data.txt` exists in the Oracle Cloud Infrastructure bucket that you can import. The sample file in this example has the following contents:

```
1,OCI Direct Sales
2,OCI Tele Sales
3,OCI Catalog
4,OCI Internet
5,OCI Partners
```

On the Oracle Cloud Infrastructure side, log in to your Oracle Cloud Infrastructure account and do the following:

1. Obtain an Auth token for the Oracle Cloud Infrastructure account.

   For more information, see Getting an Auth Token.
2. From the Object Details page, obtain the object URL for the data file stored in the Oracle Cloud Infrastructure Object Storage bucket.

   For more information, see Getting an Object Storage Object's Details.

Steps for Loading Data from Oracle Cloud Infrastructure

Run these steps to load data from Oracle Cloud Infrastructure Object Storage to Autonomous Database.

1. Store the Oracle Cloud Infrastructure account credentials in your Autonomous Database and specify a credential name. This enables the database to authenticate with your Oracle Cloud Infrastructure account and access the items in the Oracle Cloud Infrastructure Object Storage bucket.

```
SET DEFINE OFF
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'OCI_CRED_NAME',
    username => 'username',
    password => 'password'
  );
END;
/
```

> **Note:**
>
> Here, the *username* is your Oracle Cloud Infrastructure user name and the *password* is your user Auth token.

For detailed information about the parameters, see CREATE_CREDENTIAL Procedure.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

Optionally, you can test the access to Oracle Cloud Infrastructure as shown in this example.

```
SELECT * FROM DBMS_CLOUD.LIST_OBJECTS('OCI_CRED_NAME', 'https://
objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o/');
```

2. Create a table in your database where you want to load the data.

```
CREATE TABLE myocitable (id NUMBER, name VARCHAR2(64));
```

3. Import data from the Oracle Cloud Infrastructure bucket to your Autonomous Database.

   Specify the table name and the Oracle Cloud Infrastructure credential name followed by the Oracle Cloud Infrastructure object URL.

```
BEGIN
      DBMS_CLOUD.COPY_DATA(
          table_name => 'myocitable',
          credential_name => 'OCI_CRED_NAME',
          file_uri_list => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/oci-data.txt',
          format => json_object('delimiter' value ',')
      );
END;
/
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.

For detailed information about the parameters, see COPY_DATA Procedure.

Moreover, you can also perform data loading into Autonomous Database from Oracle Cloud Infrastructure Object Storage using UI options. For more information, see Create an OCI Cloud Store Location.

You have successfully imported data from Oracle Cloud Infrastructure Object Storage to your Autonomous Database. You can run this statement and verify the data in your table.

```
SELECT * FROM myocitable;


ID  NAME
--  --------------
 1  OCI Direct Sales
 2  OCI Tele Sales
 3  OCI Catalog
 4  OCI Internet
 5  OCI Partners
```

For more information about loading data, see Load Data from Files in the Cloud.

Load Data into Autonomous Database from AWS S3

This example shows you how to load data from Amazon S3 object storage to Autonomous Database using SQL commands.

This example uses AWS access credentials for user account authentication and an object URL for accessing the object in Amazon S3 bucket.

To load data from an Amazon S3 bucket:

1. Create credentials for AWS user account in the Autonomous Database.
2. Copy data from the Amazon S3 bucket to the database.

Topics

- Prepare for Loading Data from AWS S3
  Verify the prerequisites and prepare for loading data from Amazon S3.
- Steps for Loading Data from AWS S3
  Run these steps to load data from Amazon S3 to Autonomous Database.

Prepare for Loading Data from AWS S3

Verify the prerequisites and prepare for loading data from Amazon S3.

Prerequisites

A data file, for example, `data.txt` exists in the Amazon S3 bucket that you can import. The sample file in this example has the following contents:

```
1,Direct Sales
2,Tele Sales
3,Catalog
```

```
4,Internet
5,Partners
```

On the AWS side, log in to your AWS account and do the following:

1. Grant access privileges to the AWS IAM user for the Amazon S3 bucket.

   For more information, see Controlling access to a bucket with user policies.

2. Create an access key for the user.

   For more information, see Managing access keys for IAM users.

3. Obtain an object URL for the data file stored in the Amazon S3 bucket.

   For more information, see Accessing and listing an Amazon S3 bucket.

Steps for Loading Data from AWS S3

Run these steps to load data from Amazon S3 to Autonomous Database.

1. Store the AWS access credentials in your Autonomous Database and specify a credential name. This enables the database to authenticate with your AWS user account and access the items in the S3 bucket.

   ```
   SET DEFINE OFF
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'AWS_CRED_NAME',
       username => 'username',
       password => 'password'
     );
   END;
   /
   ```

   > **Note:**
   >
   > Here, the *username* is your AWS Access key ID and the *password* is your user access key.

   For detailed information about the parameters, see CREATE_CREDENTIAL Procedure.

   Creating a credential to access AWS resources is not required if you enable Amazon Resource Names (ARNs). See Use Amazon Resource Names (ARNs) to Access AWS Resources for more information.

   Optionally, you can test the access to S3 bucket as shown in this example.

   ```
   SELECT * FROM DBMS_CLOUD.LIST_OBJECTS('AWS_CRED_NAME', ' https://aws-
   bucket-01.s3.amazonaws.com/');
   ```

2. Create a table in your database where you want to load the data.

   ```
   CREATE TABLE mytable (id NUMBER, name VARCHAR2(64));
   ```

3. Import data from the Amazon S3 bucket to your Autonomous Database.

Specify the table name and the AWS credential name followed by the S3 object URL.

```
BEGIN
      DBMS_CLOUD.COPY_DATA(
          table_name => 'mytable',
          credential_name => 'AWS_CRED_NAME',
          file_uri_list => https://aws-bucket-01.s3.amazonaws.com/
data.txt',
          format => json_object('delimiter' value ',')
      );
END;
/
```

For detailed information about the parameters, see COPY_DATA Procedure.

Moreover, you can also perform data loading into Autonomous Database from Amazon S3 using UI options. For more information, see Create an Amazon S3 Cloud Store Location.

You have successfully imported data from Amazon S3 to your Autonomous Database. You can run this statement and verify the data in your table.

```
SELECT * FROM mytable;


ID  NAME
--  --------------
 1  Direct Sales
 2  Tele Sales
 3  Catalog
 4  Internet
 5  Partners
```

For more information about loading data, see Load Data from Files in the Cloud.

Load Data into Autonomous Database from Azure Blob Storage

This example shows you how to load data from Azure Blob Storage to Autonomous Database using SQL commands.

This example uses Azure access credentials for user account authentication and an object URL for accessing the object in your Azure Storage account container.

To load data from Azure Blob Storage:

1. Create credentials for Azure user account in the Autonomous Database.
2. Copy data from Azure Blob Storage to the database.

Topics

- Prepare for Loading Data from Azure Blob Storage
  Verify the prerequisites and prepare for loading data from Azure Blob Storage.
- Steps for Loading Data from Azure Blob Storage
  Run these steps to load data from Azure Blob Storage to Autonomous Database.

Prepare for Loading Data from Azure Blob Storage

Verify the prerequisites and prepare for loading data from Azure Blob Storage.

Prerequisites

A data file, for example, `azure-data.txt` exists in the Azure Storage account container that you can import. The sample file in this example has the following contents:

```
1,Azure Direct Sales
2,Azure Tele Sales
3,Azure Catalog
4,Azure Internet
5,Azure Partners
```

On the Azure side, log in to your Azure Storage account and do the following:

1. Grant the required role, for example Storage Blob Data Contributor, to your Azure Storage account.

   For more information, see Assign Azure roles using the Azure portal.

2. Obtain an access key for the Azure Storage account.

   For more information, see View account access keys.

3. Obtain the object URL for the data file stored in the Azure Storage account container.

Steps for Loading Data from Azure Blob Storage

Run these steps to load data from Azure Blob Storage to Autonomous Database.

1. Store the Azure Storage account credentials in your Autonomous Database and specify a credential name. This enables the database to authenticate with your Azure Storage account and access the items in the Azure Storage account container.

   ```
   SET DEFINE OFF
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'AZURE_CRED_NAME',
       username => 'username',
       password => 'password'
     );
   END;
   /
   ```

   > **Note:**
   >
   > Here, the *username* is your Azure Storage account name and the *password* is your user access key.

   For detailed information about the parameters, see CREATE_CREDENTIAL Procedure.

   Creating a credential to access Azure Blob Storage is not required if you enable Azure service principal. See Use Azure Service Principal to Access Azure Resources for more information.

Optionally, you can test the access to Azure Blob Storage as shown in this example.

```
SELECT * FROM DBMS_CLOUD.LIST_OBJECTS('AZURE_CRED_NAME', 'https://
dbazure.blob.core.windows.net/my-azure-container/');
```

2. Create a table in your database where you want to load the data.

```
CREATE TABLE myazuretable (id NUMBER, name VARCHAR2(64));
```

3. Import data from the Azure Blob Storage container to your Autonomous Database.

   Specify the table name and the Azure credential name followed by the Azure Blob Storage object URL.

```
BEGIN
      DBMS_CLOUD.COPY_DATA(
          table_name => 'myazuretable',
          credential_name => 'AZURE_CRED_NAME',
          file_uri_list => 'https://dbazure.blob.core.windows.net/my-azure-
container/azure-data.txt',
          format => json_object('delimiter' value ',')
      );
END;
/
```

   For detailed information about the parameters, see COPY_DATA Procedure.

   Moreover, you can also perform data loading into Autonomous Database from Azure Blob Storage using UI options. For more information, see Create an Microsoft Azure Cloud Store Location.

You have successfully imported data from Azure Blob Storage to your Autonomous Database. You can run this statement and verify the data in your table.

```
SELECT * FROM myazuretable;


ID  NAME
--  --------------
 1  Azure Direct Sales
 2  Azure Tele Sales
 3  Azure Catalog
 4  Azure Internet
 5  Azure Partners
```

For more information about loading data, see Load Data from Files in the Cloud.

Load a Fixed-Width File into a New Table

This provides an example using `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` to load data from a fixed-width source file to an external table.

For this example, the fixed-width source file has the following data:

```
  0INDEX01    INDEX                  2001272020012720200127VALID
  1INDEX02    INDEX                  2001272020012720200127VALID
  2INDEX03    INDEX                  2001272020012720200127VALID
```

```
 3INDEX04    INDEX                 2001272020012720200127VALID
 4TABLE01    TABLE                 2001272020012720200918VALID
 5TABLE02    TABLE                 2001272020012720200918VALID
 6CLUSTER01  CLUSTER               2001272020012720200127VALID
 7INDEX05    INDEX                 2001272020012720200127VALID
 8INDEX06    INDEX                 2001272020012720200127VALID
 9INDEX07    INDEX                 2001272020012720200127VALID
10INDEX08    INDEX                 2001272020012720200127VALID
11TABLE03    TABLE                 2001272020012720200127VALID
12INDEX09    INDEX                 2001272020012720200127VALID
13INDEX10    INDEX                 2001272020012720200127VALID
14TABLE04    TABLE                 2001272020012720200127VALID
15INDEX11    INDEX                 2001272020012720200127VALID
```

1. From the console, select the compartment for your Autonomous Database, and then select the link to your Autonomous Database to open the console.

> **Note:**
>
> These steps are shown using **Database Actions** to execute the PL/SQL code, and query the data. These actions can be performed from any SQL connection, connecting to the Autonomous Database as a user with the proper privileges.

2. On the Autonomous Database Details page select **Database Actions** and in the list click **SQL**.

   As an alternative, select **Database Actions** and click **View all database actions** to access the Database Actions Launchpad. From the **Development** section of the **Database Actions Launchpad**, select **SQL**.

3. Within the SQL Worksheet, enter and execute the following code:

```
BEGIN DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
             table_name      => '<YOUR_TABLE_NAME>'
,            credential_name => '<YOUR_CREDENTIAL_NAME>'
,            file_uri_list   => '<YOUR_ORACLE_OBJECT_STORE_URL>'
,            format          => json_object('trimspaces' value
'rtrim','skipheaders' value '1', 'dateformat' value 'YYYYMMDD')
,            field_list      => 'object_id      (1:3)   char
                               , object_name    (4:14)  char
                               , object_type    (15:39)  char
                               , created_date1  (40:45)  date mask "YYMMDD"
                               , created_date2  (46:53)  date
                               , last_ddl_time  (54:61)  date
                               , status         (62:71)'
,            column_list     => 'object_id      number
                               , object_name    varchar2(30)
                               , object_type    varchar2(25)
                               , status         varchar2(10)
                               , created_date1  date
                               , created_date2  date
                               , last_ddl_time  date');
END;
/
```

Import Data Using Oracle Data Pump on Autonomous Database

Oracle Data Pump offers very fast bulk data and metadata movement between Oracle databases and Autonomous Databases.

Data Pump Import lets you import data from Data Pump files residing on Oracle Cloud Infrastructure Object Storage, Microsoft Azure, AWS S3, and Oracle Cloud Infrastructure Object Storage Classic. You can save your data to your Cloud Object Store and use Oracle Data Pump to load data to Autonomous Database.

When a load or import operation results in the following timezone related error, you need to get your timezone file upgraded to the latest version available for your database:

```
ORA-39405: Oracle Data Pump does not support importing from a source database
with TSTZ version n+1
into a target database with TSTZ version n.
```

See Manage Time Zone File Updates on Autonomous Database for more information on this timezone related error.

- Export Your Existing Oracle Database to Import into Autonomous Database
  Use Oracle Data Pump Export to export your existing Oracle Database to migrate to Autonomous Database using Oracle Data Pump Import.
- Import Data Using Oracle Data Pump Version 18.3 or Later
  Oracle recommends using the latest Oracle Data Pump version for importing data from Data Pump files into your Autonomous Database, as it contains enhancements and fixes for a better experience.
- Import Data Using Oracle Data Pump (Versions 12.2.0.1 and Earlier)
  You can import data from Data Pump files into your Autonomous Database using Data Pump client versions 12.2.0.1 and earlier by setting the `default_credential` parameter.
- Access Log Files for Data Pump Import
  The log files for Data Pump Import operations are stored in the directory you specify with the data pump `impdp` *directory* parameter.
- Oracle Data Pump Import and Table Compression
  Provides notes for using Oracle Data Pump import on Autonomous Database.

Export Your Existing Oracle Database to Import into Autonomous Database

Use Oracle Data Pump Export to export your existing Oracle Database to migrate to Autonomous Database using Oracle Data Pump Import.

Oracle recommends using Oracle Data Pump schema mode to migrate your database to Autonomous Database. You can list the schemas you want to export by using the *schemas* parameter.

For a faster migration, export your schemas into multiple Data Pump files and use parallelism. You can specify the dump file name format you want to use with the *dumpfile* parameter. Set the *parallel* parameter to at least the number of CPUs you have in your database.

Oracle recommends using the following Data Pump parameters for faster and easier migration to Autonomous Database:

```
exclude=cluster,indextype,db_link
parallel=n
```

```
schemas=schema_name
dumpfile=export%u.dmp
```

The `exclude` parameters ensure that these object types are not exported.

With `encryption_pwd_prompt=yes` Oracle Data Pump export prompts for an encryption password to encrypt the dump files.

The following example exports the SH schema from a source Oracle Database for migration to a database with 16 CPUs:

```
expdp sh/sh@orcl \
exclude=cluster,indextype,db_link \
parallel=16 \
schemas=sh \
dumpfile=export%u.dmp \
encryption_pwd_prompt=yes
```

> **Note:**
>
> If during the export with `expdp` you use the `encryption_pwd_prompt=yes` parameter then also use `encryption_pwd_prompt=yes` with your import and input the same password at the `impdp` prompt to decrypt the dump files (remember the password you supply during export). The maximum length of the encryption password is 128 bytes.

You can use other Data Pump Export parameters, like compression, depending on your requirements. For more information on Oracle Data Pump Export see *Oracle Database Utilities*.

Import Data Using Oracle Data Pump Version 18.3 or Later

Oracle recommends using the latest Oracle Data Pump version for importing data from Data Pump files into your Autonomous Database, as it contains enhancements and fixes for a better experience.

Download the latest version of Oracle Instant Client, which includes Oracle Data Pump, for your platform from Oracle Instant Client Downloads. See the installation instructions on the platform install download page for the installation steps required after you download Oracle Instant Client.

In Oracle Data Pump version 18.3 and later, the *credential* argument authenticates Data Pump to the Cloud Object Storage service you are using for your source files. The *dumpfile* argument is a comma delimited list of URLs for your Data Pump files.

In Oracle Data Pump, if your source files reside on Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure native URIs or Swift URIs. See DBMS_CLOUD URI Formats for details on these file URI formats.

Importing with Oracle Data Pump and Setting `credential` Parameter

1. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example, to create Oracle Cloud Infrastructure Auth Token credentials:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DEF_CRED_NAME',
   ```

```
    username => 'adb_user@example.com',
    password => 'password'
  );
END;
/
```

For more information on Oracle Cloud Infrastructure Auth Token authentication see CREATE_CREDENTIAL Procedure.

For example, to create Oracle Cloud Infrastructure Signing Key based credentials:

```
BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL (
       credential_name => 'DEF_CRED_NAME',
       user_ocid       =>
'ocid1.user.oc1..aaaaaaaauq54mi7zdyfhw33ozkwuontjceel7fok5nq3bf2vwetkpqsoa'
,
       tenancy_ocid    =>
'ocid1.tenancy.oc1..aabbbbbbaafcue47pqmrf4vigneebgbcmmoy5r7xvoypicjqqge32ew
nrcyx2a',
       private_key     =>
'MIIEogIBAAKCAQEAtUnxbmrekwgVac6FdWeRzoXvIpA9+0r1.....wtnNpESQQQ0QLGPD8NM//
JEBg=',
       fingerprint     =>
'f2:db:f9:18:a4:aa:fc:94:f4:f6:6c:39:96:16:aa:27');
END;
/
```

For more information on Oracle Cloud Infrastructure Signing Key based credentials see CREATE_CREDENTIAL Procedure.

Supported credential types:

- Data Pump Import supports Oracle Cloud Infrastructure Auth Token based credentials and Oracle Cloud Infrastructure Signing Key based credentials.

  For more information on the credential types for Oracle Cloud Infrastructure Cloud Object Storage, see CREATE_CREDENTIAL Procedure.

- Data Pump supports using an Oracle Cloud Infrastructure Object Storage pre-authenticated URL for the `dumpfile` parameter. When you use a pre-authenticated URL, providing the `credential` parameter is required and `impdp` ignores the `credential` parameter. When you use a pre-authenticated URL for the `dumpfile`, you can use a `NULL` value for the `credential` in the next step. See Using Pre-Authenticated Requests for more information.

- Data Pump supports using a resource principal credential with `impdp`. See Import Data Using Oracle Data Pump with OCI Resource Principal Credential for more information.

2. Run Data Pump Import with the *dumpfile* parameter set to the list of file URLs on your Cloud Object Storage and the *credential* parameter set to the name of the credential you created in the previous step. For example:

```
impdp admin/password@db2022adb_high \
     directory=data_pump_dir \
     credential=def_cred_name \
     dumpfile= https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/export%u.dmp \
     parallel=16 \
```

```
encryption_pwd_prompt=yes \
exclude=cluster,indextype,db_link
```

> **Note:**
>
> If during the export with `expdp` you used the `encryption_pwd_prompt=yes` parameter then use `encryption_pwd_prompt=yes` and input the same password at the `impdp` prompt that you specified during the export.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

The `credential` parameter cannot be an Azure service principal, Amazon Resource Name (ARN), or a Google service account. See Configure Policies and Roles to Access Resources for more information on resource principal based authentication.

When you use a pre-authenticated URL for the `dumpfile`, you can use a `NULL` value for the `credential`.

For the best import performance use the `HIGH` database service for your import connection and set the `parallel` parameter to one quarter the number of ECPUs (.25 x **ECPU count**). If you are using OCPU compute model, set the parallel parameter to the number of OCPUs (1 x **OCPU count**).

For information on which database service name to connect to run Data Pump Import, see Manage Concurrency and Priorities on Autonomous Database.

For the dump file URL format for different Cloud Object Storage services, see DBMS_CLOUD URI Formats.

In this example the following are excluded during the Data Pump Import:

- Clusters
- Indextypes
- Database links

To perform a full import or to import objects that are owned by other users, you need the `DATAPUMP_CLOUD_IMP` role.

You can also use Data Pump Import to import SODA collections on Autonomous Database. See Import SODA Collection Data Using Oracle Data Pump Version 19.6 or Later for more information.

For information on disallowed objects in Autonomous Database, see SQL Commands.

See Oracle Data Pump Import and Table Compression for details on table compression using Oracle Data Pump import on Autonomous Database.

For detailed information on Oracle Data Pump Import parameters see *Oracle Database Utilities*.

Notes for importing with Oracle Data Pump:

- When you perform Oracle Data Pump export to Object Storage with a Swift URI you must use Swift credentials to import with Oracle Data Pump import. See Oracle Cloud Infrastructure Object Storage Swift URI Format for more information on Swift URIs.

- When you perform Oracle Data Pump export to Object Storage with a native URI, you can import using either Swift credentials or Signing Key based Credentials. See Oracle Cloud Infrastructure Object Storage Native URI Format for more information on Native URIs.
- Import Data Using Oracle Data Pump with OCI Resource Principal Credential
  Oracle Data Pump supports importing data pump files into your Autonomous Database using an Oracle Cloud Infrastructure resource principal as a credential object.

Import Data Using Oracle Data Pump with OCI Resource Principal Credential

Oracle Data Pump supports importing data pump files into your Autonomous Database using an Oracle Cloud Infrastructure resource principal as a credential object.

If you use Oracle Data Pump `expdp` to export directly to Object Store then you must use the same credential that was used to export when you import with `impdp`. In this case, Oracle Data Pump import does not support Oracle Cloud Infrastructure resource principal credentials. Other methods for uploading are supported for using `impdp` using resource principal credentials. For example, if you upload Oracle Data Pump files on Object Store using `DBMS_CLOUD.PUT_OBJECT`, you can import the files using Oracle Data pump `impdp` using resource principal credentials. Likewise, when you use the Oracle Cloud Infrastructure Console to upload data pump files to Object Store, you can use resource principal credentials to import into an Autonomous Database instance with Oracle Data pump `impdp`.

In Oracle Data Pump, if your source files reside on Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure native URIs or Swift URIs. See DBMS_CLOUD URI Formats for details on these file URI formats.

1. Configure the dynamic groups and policies and enable Oracle Cloud Infrastructure resource principal to access the Object Store location where the data pump files you want to import reside.

   See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

2. Run Data Pump Import with the *dumpfile* parameter set to the list of file URLs on your Cloud Object Storage and the *credential* parameter set to `OCI$RESOURCE_PRINCIPAL`.

   For example:

```
impdp admin/password@db2022adb_high \
     directory=data_pump_dir \
     credential= 'OCI$RESOURCE_PRINCIPAL' \
     dumpfile= https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/export%u.dmp \
     parallel=16 \
     encryption_pwd_prompt=yes \
     exclude=cluster,indextype,db_link
```

> **Note:**
>
> If during the export with `expdp` you used the `encryption_pwd_prompt=yes` parameter then use `encryption_pwd_prompt=yes` and input the same password at the `impdp` prompt that you specified during the export.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

For the best import performance use the `HIGH` database service for your import connection and set the `parallel` parameter to one quarter the number of ECPUs (.25 x **ECPU count**). If you are using OCPU compute model, set the parallel parameter to the number of OCPUs (1 x **OCPU count**).

For information on which database service name to connect to run Data Pump Import, see Manage Concurrency and Priorities on Autonomous Database.

For the dump file URL format for different Cloud Object Storage services, see DBMS_CLOUD URI Formats.

In this example the following are excluded during the Data Pump Import:

- Clusters
- Indextypes
- Database links

To perform a full import or to import objects that are owned by other users, you need the `DATAPUMP_CLOUD_IMP` role.

You can also use Data Pump Import to import SODA collections on Autonomous Database. See Import SODA Collection Data Using Oracle Data Pump Version 19.6 or Later for more information.

For information on disallowed objects in Autonomous Database, see SQL Commands.

For detailed information on Oracle Data Pump Import parameters see *Oracle Database Utilities*.

Import Data Using Oracle Data Pump (Versions 12.2.0.1 and Earlier)

You can import data from Data Pump files into your Autonomous Database using Data Pump client versions 12.2.0.1 and earlier by setting the `default_credential` parameter.

Data Pump Import versions 12.2.0.1 and earlier do not have the *credential* parameter. If you are using an older version of Data Pump Import you need to define a default credential property for Autonomous Database and use the `default_credential` keyword in the `dumpfile` parameter.

In Oracle Data Pump, if your source files reside on Oracle Cloud Infrastructure Object Storage you can use the Oracle Cloud Infrastructure native URIs, or Swift URIs. See DBMS_CLOUD URI Formats for details on these file URI formats.

Importing with Older Oracle Data Pump Versions and Setting `default_credential`

1. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example, to create Oracle Cloud Infrastructure Auth Token credentials:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adb_user@example.com',
    password => 'password'
  );
END;
/
```

For more information on Oracle Cloud Infrastructure Auth Token authentication see CREATE_CREDENTIAL Procedure.

For example, to create Oracle Cloud Infrastructure Signing Key based credentials:

```
BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL (
       credential_name => 'DEF_CRED_NAME',
       user_ocid       =>
'ocid1.user.oc1..aaaaaaaauq54mi7zdyfhw33ozkwuontjceel7fok5nq3bf2vwetkpqsoa'
,
       tenancy_ocid    =>
'ocid1.tenancy.oc1..aabbbbbbaafcue47pqmrf4vigneebgbcmmoy5r7xvoypicjqqge32ew
nrcyx2a',
       private_key     =>
'MIIEogIBAAKCAQEAtUnxbmrekwgVac6FdWeRzoXvIpA9+0r1.....wtnNpESQQQ0QLGPD8NM//
JEBg=',
       fingerprint     =>
'f2:db:f9:18:a4:aa:fc:94:f4:f6:6c:39:96:16:aa:27');
END;
/
```

For more information on Oracle Cloud Infrastructure Signing Key based credentials see CREATE_CREDENTIAL Procedure.

Supported credential types:

- Data Pump import supports Oracle Cloud Infrastructure Auth Token based credentials and Oracle Cloud Infrastructure Signing Key based credentials.

  For more information on the credential types for Oracle Cloud Infrastructure Cloud Object Storage, see CREATE_CREDENTIAL Procedure.

- Data Pump supports using an Oracle Cloud Infrastructure Object Storage pre-authenticated URL for the `dumpfile`. When you use a pre-authenticated URL, setting the `DEFAULT_CREDENTIAL` is required and `impdp` ignores the `DEFAULT_CREDENTIAL`. When you use a pre-authenticated URL for the `dumpfile`, you can use a `NULL` value for the `DEFAULT_CREDENTIAL` you set in the next step. See Using Pre-Authenticated Requests for more information.

- Data Pump supports using a resource principal credential with `impdp`.

2. Set the credential as the default credential for your Autonomous Database, as the ADMIN user. For example:

```
ALTER DATABASE PROPERTY SET DEFAULT_CREDENTIAL = 'ADMIN.DEF_CRED_NAME'
```

The `DEFAULT_CREDENTIAL` can be an OCI Resource Principal. For example:

```
ALTER DATABASE PROPERTY SET DEFAULT_CREDENTIAL = 'OCI$RESOURCE_PRINCIPAL'
```

See Configure Policies and Roles to Access Resources for more information on resource principal based authentication.

> **Note:**
>
> The `DEFAULT_CREDENTIAL` value cannot be an Azure service principal, Amazon Resource Name (ARN), or a Google service account.

The `DEFAULT_CREDENTIAL` value can be set to `NULL` if you are using a pre-authenticated URL.

3. Run Data Pump Import with the `dumpfile` parameter set to the list of file URLs on your Cloud Object Storage, and set the `default_credential` keyword. For example:

```
impdp admin/password@db2022adb_high \
     directory=data_pump_dir \
     dumpfile=default_credential:https://objectstorage.us-
ashburn-1.oraclecloud.com/n/namespace-string/b/bucketname/o/export%u.dmp \
     parallel=16 \
     encryption_pwd_prompt=yes \
     exclude=cluster,indextype,db_link
```

> **Note:**
>
> If during the export with `expdp` you used the `encryption_pwd_prompt=yes` parameter then use `encryption_pwd_prompt=yes` and input the same password at the `impdp` prompt that you specified during the export.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

For the best import performance use the `HIGH` database service for your import connection and set the `parallel` parameter to one quarter the number of ECPUs (.25 x **ECPU count**). If you are using OCPU compute model, set the parallel parameter to the number of OCPUs (1 x **OCPU count**).

For information on which database service name to connect to run Data Pump Import, Manage Concurrency and Priorities on Autonomous Database.

For the dump file URL format for different Cloud Object Storage services, see DBMS_CLOUD URI Formats.

In this example the following are excluded during the Data Pump Import:

- Clusters
- Indextypes
- Database links

> **Note:**
>
> To perform a full import or to import objects that are owned by other users, you need the `DATAPUMP_CLOUD_IMP` role.

You can also use Data Pump Import to import SODA collections on Autonomous Database. See Import SODA Collection Data Using Oracle Data Pump Version 19.6 or Later for more information.

For information on disallowed objects in Autonomous Database, see SQL Commands.

See Oracle Data Pump Import and Table Compression for details on table compression using Oracle Data Pump import on Autonomous Database.

For detailed information on Oracle Data Pump Import parameters see *Oracle Database Utilities*.

Notes for importing with Oracle Data Pump:

- When you perform Oracle Data Pump export to Object Storage with a Swift URI you must use Swift credentials to import with Oracle Data Pump import. See Oracle Cloud Infrastructure Object Storage Swift URI Format for more information on Swift URIs.
- When you perform Oracle Data Pump export to Object Storage with a native URI, you can import using either Swift credentials or Signing Key based Credentials. See Oracle Cloud Infrastructure Object Storage Native URI Format for more information on Native URIs.

Access Log Files for Data Pump Import

The log files for Data Pump Import operations are stored in the directory you specify with the data pump `impdp` *directory* parameter.

To access the log file you need to move the log file to your Cloud Object Storage using the procedure `DBMS_CLOUD.PUT_OBJECT`. For example, the following PL/SQL block moves the file `import.log` to your Cloud Object Storage:

```
BEGIN
  DBMS_CLOUD.PUT_OBJECT(
    credential_name => 'DEF_CRED_NAME',
    object_uri => 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/import.log',
    directory_name  => 'DATA_PUMP_DIR',
    file_name => 'import.log');
END;
/
```

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

For more information, see DBMS_CLOUD Subprograms and REST APIs.

Oracle Data Pump Import and Table Compression

Provides notes for using Oracle Data Pump import on Autonomous Database.

By default the Oracle Data Pump Import utility imports data with the same compression type as specified for tables on the source database (the database where you exported your data from).

If you want to leave compression to Autonomous Database, specify the following parameter when you import your data:

```
TRANSFORM=TABLE_COMPRESSION_CLAUSE:NONE
```

The `TRANSFORM` parameter with this option specifies that Oracle Data Pump Import should ignore the compression type of your source tables. Using this option Oracle Data Pump imports the tables into Autonomous Database using the default compression type, where the default compression type depends on the Autonomous Database workload type:

- **Data Warehouse**: The default table compression is Hybrid Columnar Compression.

  Oracle recommends using this default if your application primarily uses bulk load operations on your tables, as the loads will compress the data. Query performance on these tables will benefit from compression as queries need to do less IO.

  If you have staging tables replicated from other systems using Oracle GoldenGate or other replication tools, or your application primarily uses row-by-row DML operations on tables, Oracle recommends keeping the tables uncompressed or using Advanced Row Compression.
- **Transaction Processing**: The default table compression is no compression.
- **JSON Database**: The default table compression is no compression.
- **APEX**: The default table compression is no compression.

See TRANSFORM for more information on the Oracle Data Pump Import `TRANSFORM` parameter.

See About Table Compression for more information.

Load Data from Local Files with Oracle Database Actions

Using Oracle Database Actions, from the Worksheet page, you can load data from local files into an existing table.

- Load Data into Existing Autonomous Database Table with Oracle Database Actions
  You can load data into an existing table in Autonomous Database with the Database Actions import from file feature.

Load Data into Existing Autonomous Database Table with Oracle Database Actions

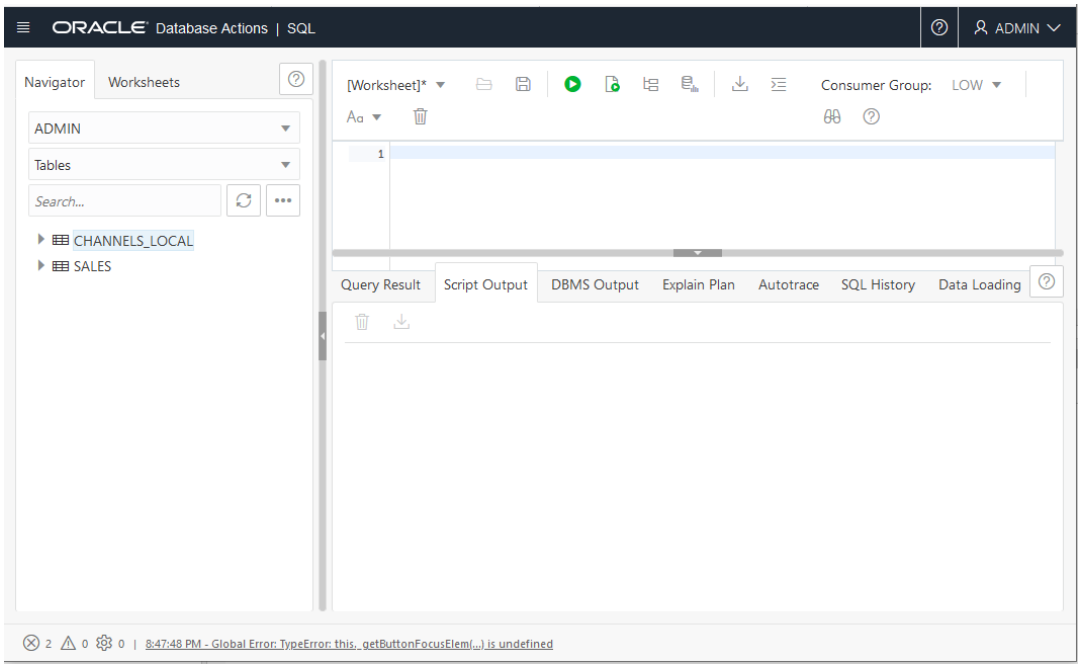
You can load data into an existing table in Autonomous Database with the Database Actions import from file feature.

Before you load data, create the table in Autonomous Database. The file formats that you can upload with the Database Actions upload feature are CSV, XLS, XLSX, TSV and TXT.

To upload data from local files to an existing table with Database Actions, do the following:

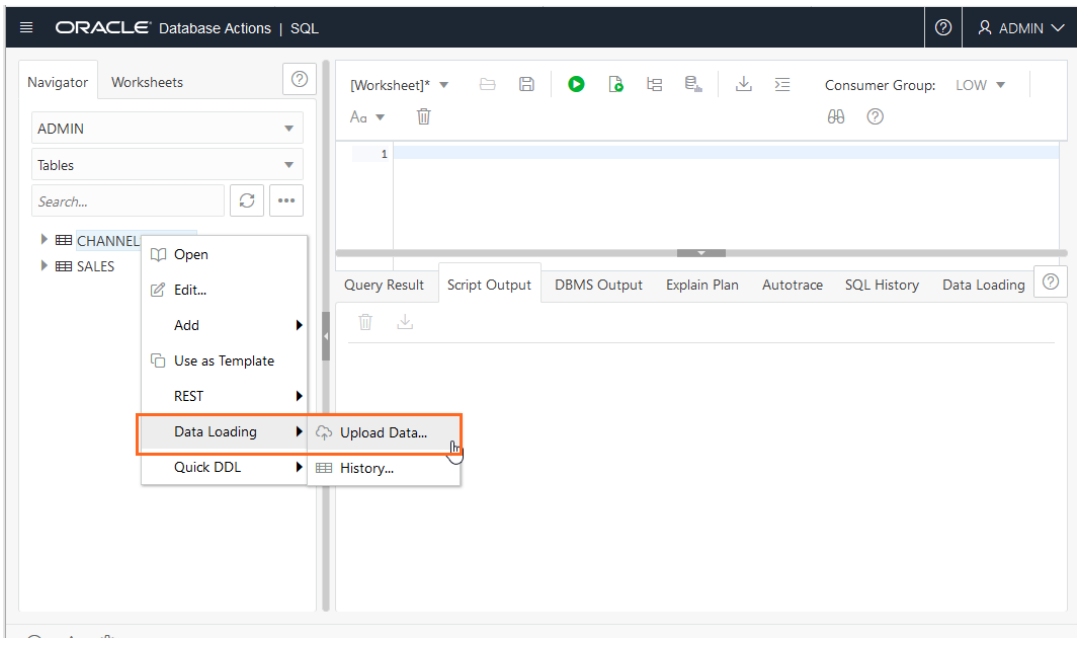
1. Access Database Actions from the Oracle Cloud Infrastructure Console or with the Database Actions link provided to you.
2. To import data, in Database Actions, under Development click **SQL**.

   This shows an SQL worksheet.

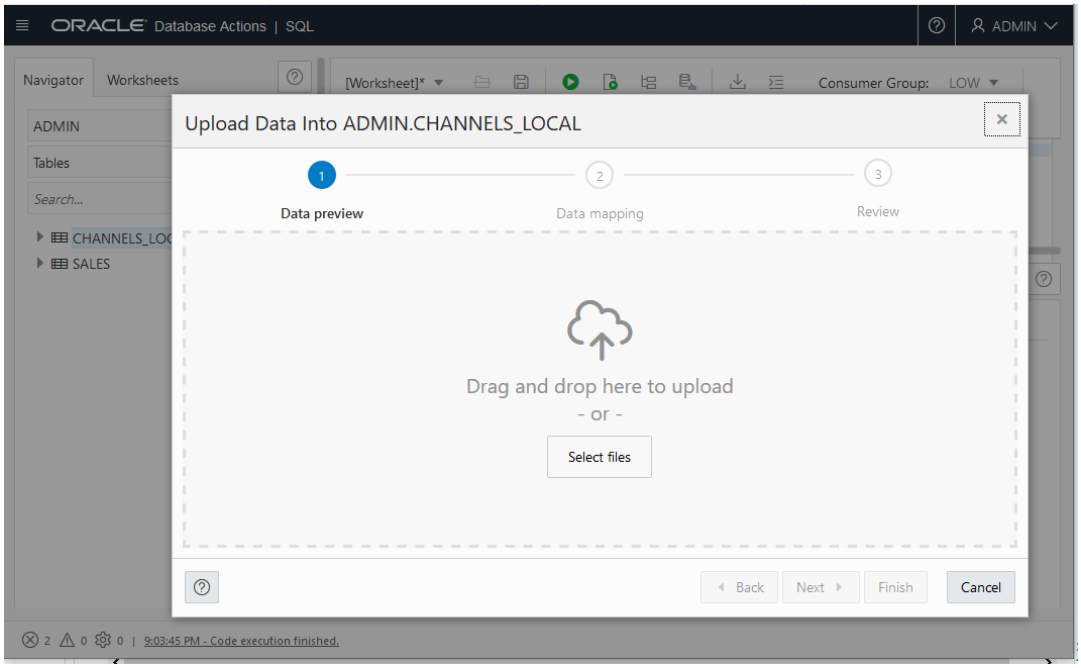


3. In the **Navigator**, right-click the table where you want to load data.
4. In the menu select **Data loading → Upload Data...**

   For example, select the `SALES` table, right-click, and select **Data loading → Upload Data...**



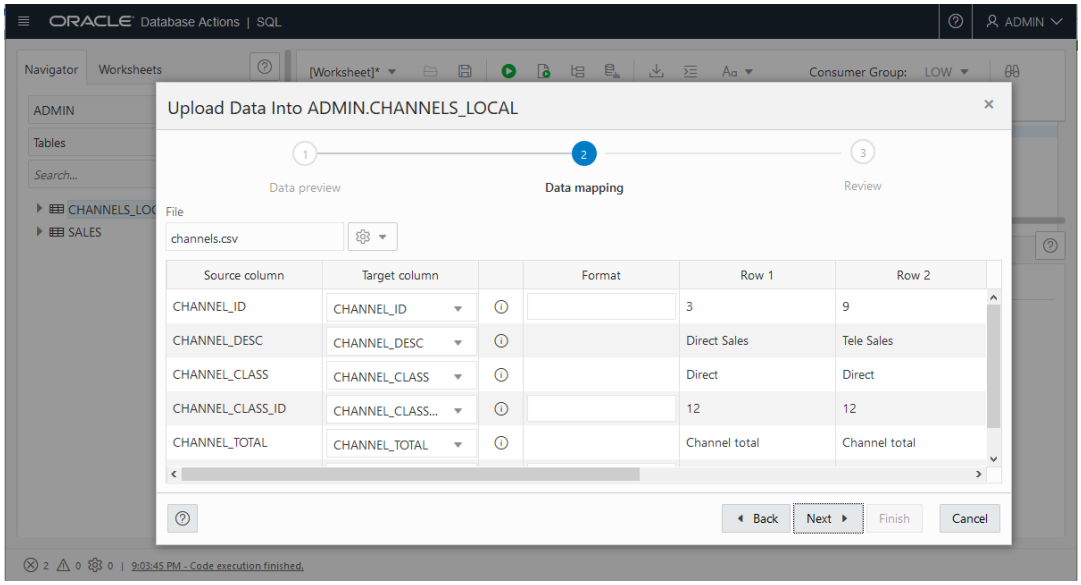
This shows the Import data dialog:



5. In the Import data dialog you can either drag and drop files or click **Select files** to show a browser to select the files to import.
6. Complete the mapping for the columns you are importing. There are a number of options for column mapping. Click (Show/Hide options) icon to show the data import and format options to change column names, skip rows, rows to load, and various other options.

   Click **Apply** to apply the options you select.
7. When you finish selecting format and mapping options, click **Next** to preview the column mapping.

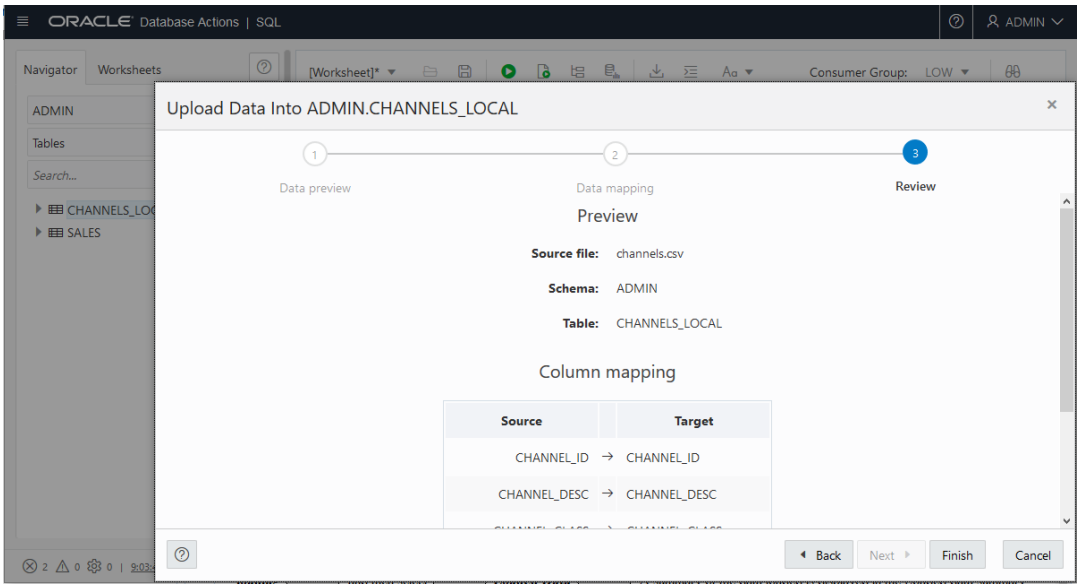
   If there is a problem at this stage, information shows with more details, such as: **2 pending actions**. This means you need to correct or fix the source file data before you import.



8. Click **Next**.

9. Click **Next** to review the column mapping.

   This shows the **Review** page to review the source columns and target columns for the import:



10. Click **Finish**.

11. Click **OK** to confirm the import.

    Depending on the size of the data file you are importing, the import may take some time.

Database Actions provides history to show the status of the import and to allow you to review the results or errors associated with the import operation.

For a detailed summary of the upload process, right-click the table in the **Navigator** tab, select **Data loading**, and then select **Loaded Data**. A summary of the data loaded is displayed in the Loaded data summary dialog.

If any data failed to load, you can view the number of rows in the Failed Rows column. Click the column and a dialog is displayed showing the failed rows.

In the Loaded data summary dialog, you can also search for files loaded by schema name, table name, or file name. To remove the loaded files, click the Delete icon.

See Uploading Data from Local Files for more information on using Database Actions to upload data.

Load Data or Query Data from Files in a Directory

You can use `DBMS_CLOUD` procedures to load data from files in a directory, including directories created on attached network file systems. You can also use these procedures to create external tables that you can use to query data.

The following `DBMS_CLOUD` procedures support loading data into the database from a directory:

- `DBMS_CLOUD.COPY_COLLECTION`
- `DBMS_CLOUD.COPY_DATA`

In addition, the following `DBMS_CLOUD` procedures support creating external tables from data in files in a directory.

- DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE
- DBMS_CLOUD.CREATE_EXTERNAL_TABLE
- DBMS_CLOUD.CREATE_HYBRID_PART_TABLE

Depending on the procedure, use either the file_uri_list parameter or the partitioning_clause parameter to specify files in one or more directories.

To load data from a directory, you do not need to include the credential_name parameter, but you need READ object privileges on the directory.

For example, you can load data into an existing table using DBMS_CLOUD.COPY_DATA:

1. Use an existing directory, create a directory, or attach a network file system for the source files.

   See Create Directory in Autonomous Database for information on creating a local directory on your Autonomous Database instance.

   See Attach Network File System to Autonomous Database for information on attaching a network file system that contains the data you want to load.

2. Load data with a DBMS_CLOUD procedure.

   For example:

```
CREATE TABLE CHANNELS
   (channel_id CHAR(1),
    channel_desc VARCHAR2(20),
    channel_class VARCHAR2(20)
   );
/

BEGIN
 DBMS_CLOUD.COPY_DATA(
    table_name => 'CHANNELS',
    file_uri_list => 'MY_DIR:channels.txt',
    format => json_object('delimiter' value ',')
 );
END;
/
```

   The parameters are:

   - table_name: is the target table's name.
   - file_uri_list: is a directory and file names specification for the source files you want to load.

     You can specify one directory and one or more file names or use a comma separated list of directories and file names. The format to specify a directory is:'*MY_DIR*:*filename.ext*'. By default the directory name *MY_DIR* is a database object and is case-insensitive. The file name is case sensitive.

     See Load Data from Directories in Autonomous Database for more details and example of specifying files in a directory. This also shows information on quoting the directory name to make the directory name case sensitive, and for information on using wildcards.

   - format: defines the options you can specify to describe the format of the source file, including whether the file is of type text, ORC, Parquet, or Avro.

See COPY_DATA Procedure for more information.

See Attach Network File System to Autonomous Database for information on attaching network file systems.

Load Data from Local Files Using SQL*Loader

Instead of using SQL*Loader Oracle recommends loading data from the Cloud Object Storage for better performance and enhanced functionality.

For information on loading from Cloud Object Store, see Load Data from Files in the Cloud.

Depending on your workload type, note the following:

- **Data Warehouse:** If you use SQL*Loader to load data, note that Autonomous Database does not gather optimizer statistics for your load and you need to gather optimizer statistics manually as explained in Manage Optimizer Statistics on Autonomous Database. Autonomous Database gathers optimizer statistics automatically for tables loaded with direct path operations issued in SQL (direct path load operations that bypass the SQL data processing, such as SQL*Loader direct path, do not collect statistics).
- **Transaction Processing** or **JSON Database**: If you use SQL*Loader to load data, note that Autonomous Database does not gather optimizer statistics for your load and you need to gather optimizer statistics manually as explained in in Manage Optimizer Statistics on Autonomous Database or wait for the automatic statistic gathering task to kick in.

For detailed information on SQL*Loader see, Oracle Database Utilities.

Using Data Pipelines for Continuous Load and Export

Data pipelines allow you to repeatedly load data from object store or export data to object store. Load pipelines provide continuous incremental data loading from external sources (as data arrives on object store it is loaded to a database table). Export pipelines provide continuous incremental data exporting to object store (as new data appears in a database table it is exported to object store).

- About Data Pipelines on Autonomous Database
- Create and Configure Pipelines
  You can create one or more load or export pipelines. When you create a pipeline, you use parameters and set pipeline attributes to configure the pipeline.
- Test Pipelines
- Control Pipelines (Start, Stop, Drop, or Reset a Pipeline)
  After you create and test a pipeline, you control a pipeline by starting, stopping, or dropping the pipeline. You can also reset a pipeline.
- Monitor and Troubleshoot Pipelines
  All pipelines that are created are logged in the `DBMS_CLOUD_PIPELINE` views.
- Use Oracle Maintained Pipelines
  Autonomous Database provides built-in pipelines. These pipelines are preconfigured and can be started by the ADMIN user. Oracle Maintained pipelines are owned by ADMIN user.

About Data Pipelines on Autonomous Database

Autonomous Database data pipelines are either load pipelines or export pipelines.

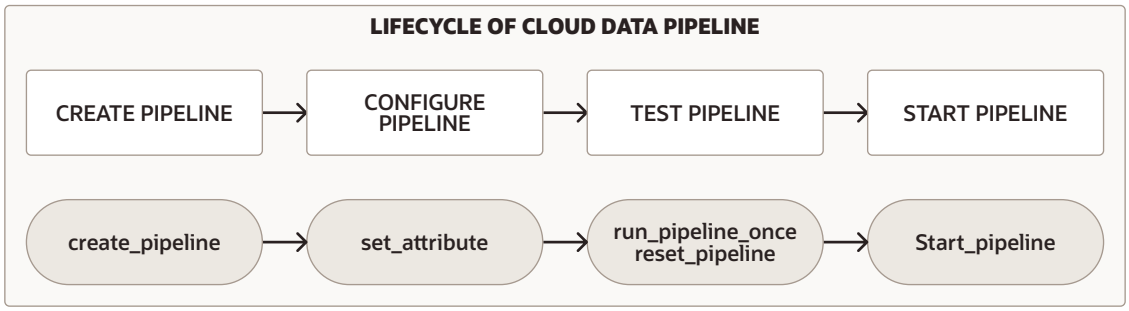
Load pipelines provide continuous incremental data loading from external sources (as data arrives on object store it is loaded to a database table). Export pipelines provide continuous incremental data exporting to object store (as new data appears in a database table it is exported to object store). Pipelines use database scheduler to continuously load or export incremental data.

Autonomous Database data pipelines provide the following:

- **Unified Operations**: Pipelines allow you to quickly and easily load or export data and repeat these operations at regular intervals for new data. The `DBMS_CLOUD_PIPELINE` package provides a unified set of PL/SQL procedures for pipeline configuration and for creating and starting a scheduled job for load or export operations.
- **Scheduled Data Processing**: Pipelines monitor their data source and periodically load or export the data as new data arrives.
- **High Performance**: Pipelines scale data transfer operations with the available resources on your Autonomous Database. Pipelines by default use parallelism for all load or export operations, and scale based on the CPU resources available on your Autonomous Database, or based on a configurable priority attribute.
- **Atomicity and Recovery**: Pipelines guarantee atomicity such that files in object store are loaded exactly once for a load pipeline.
- **Monitoring and Troubleshooting**: Pipelines provide detailed log and status tables that allow you to monitor and debug pipeline operations.
- **Multicloud Compatible**: Pipelines on Autonomous Database support easy switching between cloud providers without application changes. Pipelines support all the credential and object store URI formats that Autonomous Database supports (Oracle Cloud Infrastructure Object Storage, Amazon S3, Azure Blob Storage, Google Cloud Storage, and Amazon S3-Compatible object stores).
- About the Data Pipeline Lifecycle on Autonomous Database
  The `DBMS_CLOUD_PIPELINE` package provides procedures for creating, configuring, testing, and starting a pipeline. The pipeline lifecycle and procedures are the same for both load and export pipelines.
- About Load Pipelines on Autonomous Database
- About Export Pipelines on Autonomous Database
  Use an export pipeline for continuous incremental export of data from the database to object store. An export pipeline periodically identifies candidate data and uploads the data to object store.
- About Oracle Maintained Pipelines
  Autonomous Database provides built-in pipelines for exporting logs to object store. These pipelines are preconfigured and can be started by the ADMIN user.

About the Data Pipeline Lifecycle on Autonomous Database

The `DBMS_CLOUD_PIPELINE` package provides procedures for creating, configuring, testing, and starting a pipeline. The pipeline lifecycle and procedures are the same for both load and export pipelines.



For either pipeline type you perform the following steps to create and use a pipeline:

1. Create and configure the pipeline. See Create and Configure Pipelines for more information.
2. Test a new pipeline. See Test Pipelines for more information.
3. Start a pipeline. See Start a Pipeline for more information.

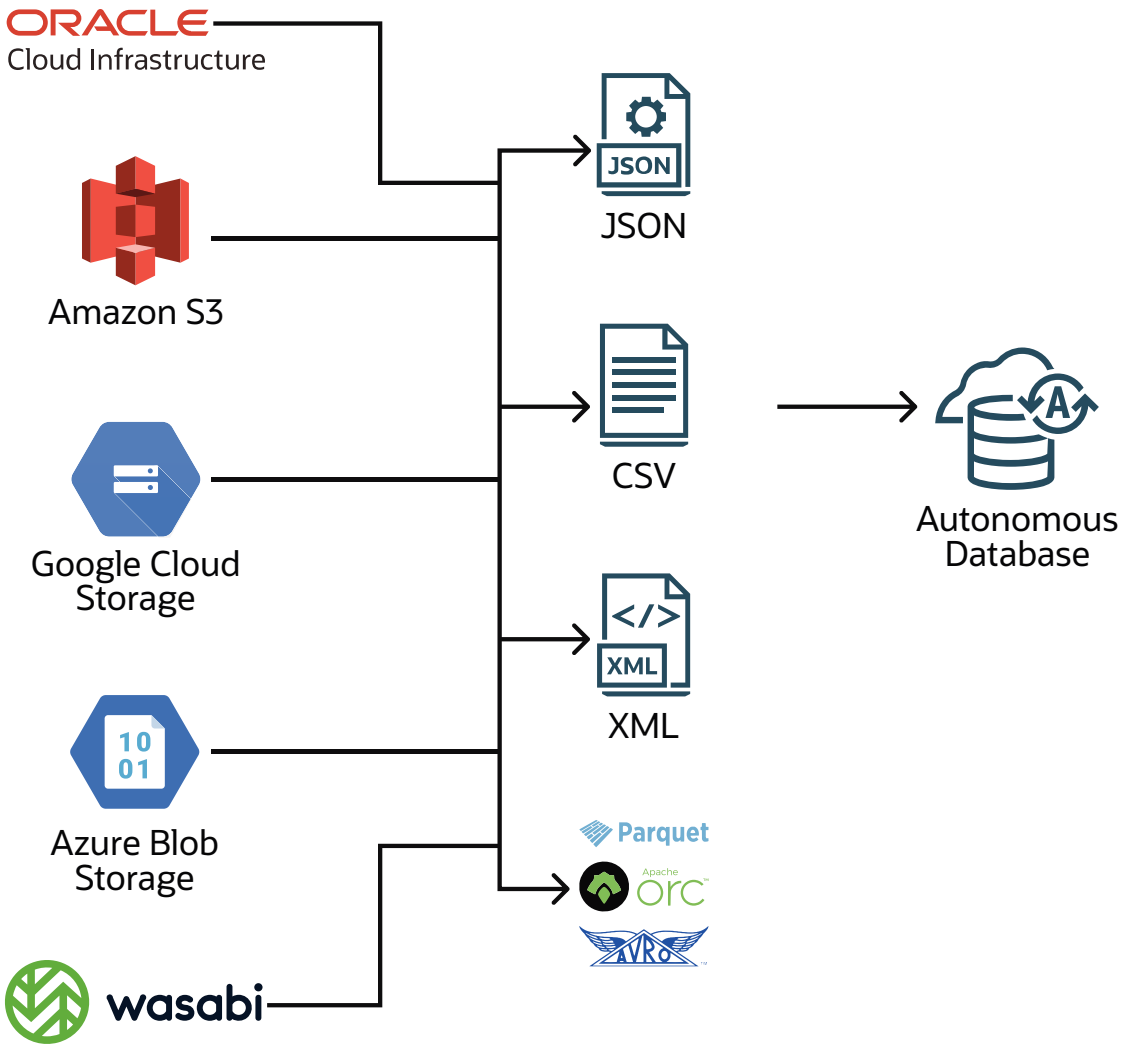
In addition, you can monitor, stop, or drop pipelines:

- While a pipeline is running, either during testing or during regular use after you start the pipeline, you can monitor the pipeline. See Monitor and Troubleshoot Pipelines for more information.
- You can stop a pipeline and later start it again, or drop a pipeline when you are finished using the pipeline. See Stop a Pipeline and Drop a Pipeline for more information.

About Load Pipelines on Autonomous Database

Use a load pipeline for continuous incremental data loading from external files in object store into a database table. A load pipeline periodically identifies new files in object store and loads the new data into the database table.
A load pipeline operates as follows (some of these features are configurable using pipeline attributes):

- Object store files are loaded in parallel into a database table.
  - A load Pipeline uses the object store file name to uniquely identify and load newer files.
  - Once a file in object store has been loaded in the database table, if the file content changes in object store, it will not be loaded again.
  - If the object store file is deleted, it does not impact the data in the database table.
- If failures are encountered, a load pipeline automatically retries the operation. Retries are attempted on every subsequent run of the pipeline's scheduled job.
- In cases where the data in a file does not comply with the database table, it is marked as `FAILED` and can be reviewed to debug and troubleshoot the issue.
  - If any file fails to load, the pipeline does not stop and continues to load the other files.
- Load pipelines support multiple input file formats, including: JSON, CSV, XML, Avro, ORC, and Parquet.



Migration from non-Oracle databases is one possible use case for a load pipeline. When you need to migrate your data from a non-Oracle database to Oracle Autonomous Database, you can extract the data and load it into Autonomous Database (Oracle Data Pump format cannot be used for migrations from non-Oracle databases). By using a generic file format such as CSV to export data from a non-Oracle database, you can save your data to files and upload the files to object store. Next, create a pipeline to load the data to Autonomous Database. Using a load pipeline to load a large set of CSV files provides important benefits such as fault tolerance, and resume and retry operations. For a migration with a large data set you can create multiple pipelines, one per table for the non-Oracle database files, to load data into Autonomous Database.

About Export Pipelines on Autonomous Database

Use an export pipeline for continuous incremental export of data from the database to object store. An export pipeline periodically identifies candidate data and uploads the data to object store.

There are three export pipeline options (the export options are configurable using pipeline attributes):

- Export incremental results of a query to object store using a date or timestamp column as key for tracking newer data.

- Export incremental data of a table to object store using a date or timestamp column as key for tracking newer data.
- Export data of a table to object store using a query to select data without a reference to a date or timestamp column (so that the pipeline exports all the data that the query selects for each scheduler run).

Export pipelines have the following features (some of these are configurable using pipeline attributes):

- Results are exported in parallel to object store.
- In case of any failures, a subsequent pipeline job repeats the export operation.
- Export pipelines support multiple export file formats, including: CSV, JSON, Parquet, or XML.

About Oracle Maintained Pipelines

Autonomous Database provides built-in pipelines for exporting logs to object store. These pipelines are preconfigured and can be started by the ADMIN user.

The Oracle Maintained pipelines are:

- `ORA$AUDIT_EXPORT`: This pipeline exports the database audit logs to object store in JSON format and runs every 15 minutes after starting the pipeline (based on the `interval` attribute value).
- `ORA$APEX_ACTIVITY_EXPORT`: This pipeline exports the Oracle APEX workspace activity log to object store in JSON format. This pipeline is preconfigured with the SQL query for retrieving APEX activity records and runs every 15 minutes after starting the pipeline (based on the `interval` attribute value).

The Oracle Maintained pipelines are owned by the ADMIN user and attributes of Oracle Maintained Pipelines can be modified by the ADMIN user.

By default the Oracle Maintained Pipelines use `OCI$RESOURCE_PRINCIPAL` as the `credential_name`.

See Use Oracle Maintained Pipelines for more information.

Create and Configure Pipelines

You can create one or more load or export pipelines. When you create a pipeline, you use parameters and set pipeline attributes to configure the pipeline.

The options to create and configure a pipeline follow:

- **Load Pipeline**:
  - See Create and Configure a Pipeline for Loading Data.
- **Export Pipeline**:
  - Export incremental results of a query to object store using a date or timestamp column as key for tracking newer data. See Create and Configure a Pipeline for Export with Timestamp Column.
  - Export incremental data of a table to object store using a date or timestamp column as key for tracking newer data. See Create and Configure a Pipeline for Export with Timestamp Column.

  - Export data of a table to object store using a query to select data without a reference to a date or timestamp column (so that the pipeline exports all the data that the query selects for each scheduler run). See Create and Configure a Pipeline to Export Query Results (Without a Timestamp).

- Create and Configure a Pipeline for Loading Data
  You can create a pipeline to load data from external files in object store to tables in Autonomous Database.
- Create and Configure a Pipeline for Export with Timestamp Column
- Create and Configure a Pipeline to Export Query Results (Without a Timestamp)
  You can create an export pipeline to automatically export data from your Autonomous Database to object store. Using this export pipeline option you specify a SQL query that the pipeline runs periodically to export data to object store. You can use this export option to share the latest data from your Autonomous Database to object store for other applications to consume the data.

Create and Configure a Pipeline for Loading Data

You can create a pipeline to load data from external files in object store to tables in Autonomous Database.

A load pipeline consumes data placed on object store and loads it to a table in Autonomous Database. When you create a load pipeline, the pipeline runs at regular intervals to consume data placed on object store, when new data files arrive the pipeline loads the new data. You can also use a pipeline to reliably copy files, with resume and retry capabilities, from object store to a table on your database.

With a load pipeline, the pipeline package uses `DBMS_CLOUD.COPY_DATA` to load data.

On your Autonomous Database, either use an existing table or create the database table where you are loading data. For example:

```
CREATE TABLE EMPLOYEE
            (name     VARCHAR2(128),
             age      NUMBER,
             salary   NUMBER);
```

1. Create a pipeline to load data from object store.

   ```
   BEGIN
        DBMS_CLOUD_PIPELINE.CREATE_PIPELINE(
           pipeline_name => 'MY_PIPE1',
           pipeline_type => 'LOAD',
           description   => 'Load metrics from object store into a table'
     );
   END;
   /
   ```

   See CREATE_PIPELINE Procedure for more information.

2. Create a credential object to access the object store that contains the files you are loading.

   You specify the credential for the pipeline source location with the attribute `credential_name`. If you do not supply a `credential_name` in the next step, the `credential_name` value is set to `NULL`. You can use the default `NULL` value when the `location` attribute is a public or pre-authenticated URL.

See CREATE_CREDENTIAL Procedure for more information.

3. Set the pipeline attributes, including the required attributes: `location`, `table_name`, and `format`.

```
BEGIN
     DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
       pipeline_name => 'MY_PIPE1',
       attributes    => JSON_OBJECT(
            'credential_name' VALUE 'OBJECT_STORE_CRED',
            'location' VALUE 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/',
            'table_name' VALUE 'employee',
            'format' VALUE '{"type":"json", "columnpath":["$.NAME",
"$.AGE", "$.SALARY"]}',
            'priority' VALUE 'HIGH',
            'interval' VALUE '20')
  );
END;
/
```

The following attributes must be set to run a load pipeline:

- `location`: Specifies the source file location on object store.
- `table_name`: Specifies the table in your database where you are loading data. The `location` you specify is for one `table_name` per pipeline.
- `format`: Describes the format of the data you are loading.

  See DBMS_CLOUD Package Format Options for more information.

The `credential_name` is the credential you created in the previous step.

The `priority` value determines the number of files loaded in parallel. A pipeline with a higher priority consumes more database resources and completes each run faster, as compared to running at a lower priority.

The `interval` value specifies the time interval in minutes between consecutive runs of a pipeline job. The default `interval` is 15 minutes.

See DBMS_CLOUD_PIPELINE Attributes for details on the pipeline attributes.

After you create a pipeline you can test the pipeline or start the pipeline:

- Test Pipelines
- Start a Pipeline

As an alternative, to set the format for JSON, you could use the following format:

```
BEGIN
    DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
        pipeline_name   => 'MY_PIPE1',
        attribute_name  => 'format',
        attribute_value => JSON_OBJECT('type' value 'json', 'columnpath'
value '["$.NAME", "$.AGE", "$.SALARY"]')
    );
END;
/
```

Create and Configure a Pipeline for Export with Timestamp Column

You can create an export pipeline to automatically export time-series data from your Autonomous Database to object store.

Using this export pipeline option you specify a table or SQL query and a column with a timestamp that the pipeline uses to keep track of the time of the last upload. You can use an export pipeline to share data for consumption by other applications or to save data to object store.

With an export pipeline, the pipeline package uses `DBMS_CLOUD.EXPORT_DATA` to export data.

An export pipeline exports data from your Autonomous Database to object store. When you create an export pipeline, the pipeline runs at regular intervals and places data on object store.

1. Create a pipeline to export data to object store.

```
BEGIN
     DBMS_CLOUD_PIPELINE.CREATE_PIPELINE(
        pipeline_name=>'EXP_PIPE1',
        pipeline_type=>'EXPORT',
        description=>'Export time series metrics to object store');
END;
/
```

   See CREATE_PIPELINE Procedure for more information.

2. Create a credential object to access the destination object store location where you are exporting data files.

   You specify the credential for the pipeline destination location with the attribute `credential_name`. If you do not supply a `credential_name` in the next step, the `credential_name` value is set to `NULL`. You can use the default `NULL` value when the `location` attribute is a public or pre-authenticated URL.

   See CREATE_CREDENTIAL Procedure for more information.

3. Set the export pipeline attributes.

   When you specify a `table_name` parameter, table rows are exported to object store. When you specify a `query` parameter, the query specifies a `SELECT` statement so that only the required data is exported to object store.

   - Using using a `table_name` parameter:

```
BEGIN
     DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
       pipeline_name => 'EXP_PIPE1',
       attributes    => JSON_OBJECT('credential_name' VALUE
'OBJECT_STORE_CRED',
          'location' VALUE 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/',
          'table_name' VALUE 'metric_table',
          'key_column' VALUE 'metric_time',
          'format' VALUE '{"type": "json"}',
          'priority' VALUE 'MEDIUM',
          'interval' VALUE '20')
  );
```

```
END;
/
```

- Using a `query` parameter:

```
BEGIN
     DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
       pipeline_name => 'EXP_PIPE1',
       attributes    => JSON_OBJECT('credential_name' VALUE
'OBJECT_STORE_CRED',
            'location' VALUE 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/',
            'query' VALUE 'SELECT * from metrics_table',
            'key_column' VALUE 'metric_time',
            'format' VALUE '{"type": "json"}',
            'priority' VALUE 'MEDIUM',
            'interval' VALUE '20')
  );
END;
/
```

Where the `credential_name` is the credential you created in the previous step.

The following attributes must be set to run an export pipeline:

- `location`: Specifies the destination object store location. The `location` you specify is for one `table_name` per pipeline.
- `table_name`: Specifies the table in your database that contains the data you are exporting (either the `table_name` parameter or the `query` parameter is required).
- `query`: Specifies the query to run in your database that provides the data you are exporting (either the `table_name` parameter or the `query` parameter is required).
- `format`: Describes the format of the data you are exporting.

  See DBMS_CLOUD Package Format Options for EXPORT_DATA for more information.

The `priority` value determines the degree of parallelism for fetching data from the database.

The `interval` value specifies the time interval in minutes between consecutive runs of a pipeline job. The default `interval` is 15 minutes.

See DBMS_CLOUD_PIPELINE Attributes for details on the pipeline attributes.

After you create a pipeline you can test the pipeline or start the pipeline:

- Test Pipelines
- Start a Pipeline

Create and Configure a Pipeline to Export Query Results (Without a Timestamp)

You can create an export pipeline to automatically export data from your Autonomous Database to object store. Using this export pipeline option you specify a SQL query that the pipeline runs periodically to export data to object store. You can use this export option to share

the latest data from your Autonomous Database to object store for other applications to consume the data.

An export pipeline exports data from your Autonomous Database to object store. When you create an export pipeline, the pipeline runs at regular intervals and places data on object store.

1. Create a pipeline to export data to object store.

```
BEGIN
     DBMS_CLOUD_PIPELINE.CREATE_PIPELINE(
        pipeline_name=>'EXP_PIPE2',
        pipeline_type=>'EXPORT',
        description=>'Export query results to object store.');
END;
/
```

   See CREATE_PIPELINE Procedure for more information.

2. Create a credential object to access the destination object store location where you are exporting data files.

   You specify the credential for the pipeline destination location with the attribute `credential_name`. If you do not supply a `credential_name` in the next step, the `credential_name` value is set to `NULL`. You can use the default `NULL` value when the `location` attribute is a public or pre-authenticated URL.

   See CREATE_CREDENTIAL Procedure for more information.

3. Set the export pipeline attributes.

```
BEGIN
     DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
       pipeline_name => 'EXP_PIPE2',
       attributes    => JSON_OBJECT(
          'credential_name' VALUE 'OBJECT_STORE_CRED',
          'location' VALUE 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/',
          'query' VALUE 'SELECT * FROM table_name',
          'format' VALUE '{"type": "json"}',
          'priority' VALUE 'MEDIUM',
          'interval' VALUE '20')
  );
END;
/
```

   Where the `credential_name` is the credential you created in the previous step.

   The following attributes must be set to run an export pipeline:

   - `location`: Specifies the destination object store location.
   - `query`: Specifies the query to run in your database that provides the data you are exporting.
   - `format`: Describes the format of the data you are exporting.

     See DBMS_CLOUD Package Format Options for EXPORT_DATA for more information.

The `priority` value determines the degree of parallelism for fetching data from the database.

The `interval` value specifies the time interval in minutes between consecutive runs of a pipeline job. The default `interval` is 15 minutes.

See DBMS_CLOUD_PIPELINE Attributes for details on the pipeline attributes.

After you create a pipeline you can test the pipeline or start the pipeline:

- Test Pipelines
- Start a Pipeline

Test Pipelines

Use `RUN_PIPELINE_ONCE` to run a pipeline once on-demand without creating a scheduled job.

`RUN_PIPELINE_ONCE` is useful for testing a pipeline before you start the pipeline. After you run a pipeline once to test the pipeline and check that it is working as expected, use `RESET_PIPELINE` to reset the pipeline's state (to the state before you ran `RUN_PIPELINE_ONCE`).

1. Create a pipeline.

   See Create and Configure Pipelines for more information.

2. Run a pipeline once to test the pipeline.

```
BEGIN
    DBMS_CLOUD_PIPELINE.RUN_PIPELINE_ONCE(
        pipeline_name => 'MY_PIPE1'
);
END;
/
```

   See RUN_PIPELINE_ONCE Procedure for more information.

3. Perform any required checks to verify the pipeline is operating as expected.

   See Monitor and Troubleshoot Pipelines for more information.

4. Reset the pipeline.

```
BEGIN
   DBMS_CLOUD_PIPELINE.RESET_PIPELINE(
     pipeline_name => 'MY_PIPE1',
     purge_data => TRUE
);
END;
/
```

   See RESET_PIPELINE Procedure for more information.

Control Pipelines (Start, Stop, Drop, or Reset a Pipeline)

After you create and test a pipeline, you control a pipeline by starting, stopping, or dropping the pipeline. You can also reset a pipeline.

- Start a Pipeline

- Stop a Pipeline
- Drop a Pipeline
- Reset a Pipeline

Start a Pipeline

After you create a pipeline, you can start the pipeline.

When a pipeline is started the pipeline runs continuously in a scheduled job. The pipeline's scheduled job repeats, either by default every 15 minutes or at the interval you set with the `interval` attribute.

1. Start a pipeline.

```
BEGIN
  DBMS_CLOUD_PIPELINE.START_PIPELINE(
      pipeline_name => 'EMPLOYEE_PIPELINE'
  );
END;
/
```

   By default, a pipeline job begins immediately, as soon as the pipeline is started. To start a pipeline job at a later time, specify a valid future date or timestamp using the `start_date` parameter.

   See START_PIPELINE Procedure for more information.

2. Verify that the pipeline is started.

   For example:

```
SELECT pipeline_name, status from USER_CLOUD_PIPELINES
   WHERE pipeline_name = 'EMPLOYEE_PIPELINE';

PIPELINE_NAME            STATUS
------------------------ -------
EMPLOYEE_PIPELINE        STARTED
```

Stop a Pipeline

Use `STOP_PIPELINE` to stop a pipeline. When a pipeline is stopped, no future jobs are scheduled for the pipeline.

By default currently running jobs complete when you stop a pipeline. Set the `force` parameter to `TRUE` to terminate any running jobs and stop the pipeline immediately.

1. Stop a pipeline.

```
BEGIN
  DBMS_CLOUD_PIPELINE.STOP_PIPELINE(
      pipeline_name => 'EMPLOYEE_PIPELINE'
  );
END;
/
```

See STOP_PIPELINE Procedure for more information.

2. Verify that the pipeline is stopped.

```
SELECT pipeline_name, status from USER_CLOUD_PIPELINES
   WHERE pipeline_name = 'EMPLOYEE_PIPELINE';

PIPELINE_NAME            STATUS
------------------------ -------
EMPLOYEE_PIPELINE        STOPPED
```

See STOP_PIPELINE Procedure for more information.

Drop a Pipeline

The procedure `DROP_PIPELINE` drops an existing pipeline.

If a pipeline has been started, it must be stopped before the pipeline can be dropped. See STOP_PIPELINE Procedure for more information.

In order to drop a pipeline that is started, set the `force` parameter to `TRUE` to terminate any running jobs and drop the pipeline immediately

1. Drop a pipeline.

```
BEGIN
  DBMS_CLOUD_PIPELINE.DROP_PIPELINE(
      pipeline_name => 'EMPLOYEE_PIPELINE'
  );
END;
/
```

2. Verify the pipeline is dropped.

```
SELECT pipeline_name, status from USER_CLOUD_PIPELINES
   WHERE pipeline_name = 'EMPLOYEE_PIPELINE';

No rows selected
```

See DROP_PIPELINE Procedure for more information.

Reset a Pipeline

Use the reset pipeline operation to clear the record of the pipeline to the initial state.

> **Note:**
>
> You can optionally use reset pipeline to purge data in the database table associated with a load pipeline or to remove files in object store for an export pipeline. Usually this option is used when you are testing a pipeline during pipeline development.

Reset pipeline operates as follows:

- **Load Pipeline**: For a load pipeline, resetting the pipeline clears the record of the files being loaded by the pipeline. When you call either `START_PIPELINE` or `RUN_PIPELINE_ONCE` after resetting a load pipeline, the pipeline repeats the data load and includes all the files present in the object store location.

  When `purge_data` is set to `TRUE`, the procedure truncates the data in the database table.

- **Export Pipeline**: For an export pipeline, resetting the pipeline clears the last tracked data in the database table. When you call either `START_PIPELINE` or `RUN_PIPELINE_ONCE` after resetting an export pipeline, the pipeline repeats exporting data from the table or query.

  When `purge_data` set to `TRUE`, the procedure deletes existing files in the object store location specified with the `location` attribute.

To reset a pipeline:

1. Stop the pipeline you want to reset.

   A data pipeline must be in stopped state to reset it. See STOP_PIPELINE Procedure for more information.

2. Reset the pipeline.

```
BEGIN
     DBMS_CLOUD_PIPELINE.RESET_PIPELINE(
        pipeline_name => 'EMPLOYEE_PIPELINE',
        purge_data => TRUE);
END;
/
```

   Only use the `purge_data` parameter with value `TRUE` if you want to clear data in your database table, for a load pipeline, or clear files in object store for an export pipeline. Usually this option is used when you are testing a pipeline during pipeline development.

   See RESET_PIPELINE Procedure for more information.

Monitor and Troubleshoot Pipelines

All pipelines that are created are logged in the `DBMS_CLOUD_PIPELINE` views.

View Pipeline Status Information

Check pipeline status and obtain other pipeline information using the `USER_CLOUD_PIPELINES` or `DBA_CLOUD_PIPELINES` views. For example, the following `SELECT` statement with a `WHERE` clause predicate on the `pipeline_name` shows that `MY_TREE_DATA` is a load pipeline and the pipeline is started:

```
SELECT pipeline_name, pipeline_type, status FROM USER_CLOUD_PIPELINES
   WHERE pipeline_name = 'MY_TREE_DATA';


PIPELINE_NAME PIPELINE_TYPE STATUS
------------- ------------- -------
MY_TREE_DATA  LOAD          STARTED
```

See DBMS_CLOUD_PIPELINE Views for more information.

View Pipeline Attributes

The pipeline attributes can be monitored by querying the `USER_CLOUD_PIPELINE_ATTRIBUTES` or `DBA_CLOUD_PIPELINE_ATTRIBUTES` views. Query these views to see pipeline attribute information.

For example:

```
SELECT pipeline_name, attribute_name, attribute_value FROM
user_cloud_pipeline_attributes
     WHERE pipeline_name = 'MY_TREE_DATA';

PIPELINE_NAME ATTRIBUTE_NAME
ATTRIBUTE_VALUE

------------- ---------------
--------------------------------------------------------------------------------
---
MY_TREE_DATA  credential_name
DEF_CRED_OBJ_STORE

MY_TREE_DATA  format          {"type":
"csv"}
MY_TREE_DATA  interval
20

MY_TREE_DATA  location        https://objectstorage.us-
ashburn-1.oraclecloud.com/n/namespace/b/treetypes/o/
MY_TREE_DATA  priority
high

MY_TREE_DATA  table_name      TREES
```

See DBMS_CLOUD_PIPELINE Views for more information.

View Pipeline History

The `USER_CLOUD_PIPELINE_HISTORY` and `DBA_CLOUD_PIPELINE_HISTORY` views show the state of running jobs. Use the pipeline history views to help you monitor the health of a pipeline and detect failures in a running pipeline.

For example:

```
SELECT pipeline_id, pipeline_name, status, error_message  FROM
user_cloud_pipeline_history
     WHERE pipeline_name = 'MY_TREE_DATA';

PIPELINE_ID PIPELINE_NAME STATUS    ERROR_MESSAGE
----------- ------------- --------- -------------
          7  MY_TREE_DATA SUCCEEDED
```

See DBMS_CLOUD_PIPELINE Views for more information.

Pipeline Status Table: Additional Monitoring for Load Pipelines

The pipeline status table shows each file name and its status for a load pipeline. The `STATUS_TABLE` column in `DBA_CLOUD_PIPELINES` and `USER_CLOUD_PIPELINES` shows the status table name.

For example, the following `SELECT` statement with a `WHERE` clause predicate on the `pipeline_name` shows the status table name for a pipeline:

```
SELECT pipeline_name, status_table FROM user_cloud_pipelines
   WHERE pipeline_name = 'MY_TREE_DATA';

PIPELINE_NAME STATUS_TABLE
------------- --------------------
MY_TREE_DATA  PIPELINE$9$41_STATUS
```

View the status table to see information about the pipeline, including the following:

- The relevant error number and error message are recorded in the status table if an operation on a specific file fails.
- For completed pipeline operations, the time needed for each operation can be calculated using the reported `START_TIME` and `END_TIME`.

For example the following shows that the load operation for two files failed and one completed:

```
SELECT id, name, status, error_code, error_message, sid FROM
PIPELINE$9$41_STATUS;

ID NAME       STATUS    ERROR_CODE ERROR_MESSAGE                      SID
-- ---------- --------- ---------- -------------------------------- -----
 1 trees1.txt FAILED         30653 ORA-30653: reject limit reached  18070
 2 trees2.txt FAILED         30653 ORA-30653: reject limit reached  18070
 3 trees3.txt COMPLETED                                             18070
```

Pipelines for loading data, where the `pipeline_type` is `'LOAD'`, reserve an `ID` that is shown in `USER_LOAD_OPERATIONS` and in `DBA_LOAD_OPERATIONS`. The `ID` value in these views maps to the pipeline's `OPERATION_ID` in `USER_CLOUD_PIPELINES` and `DBA_CLOUD_PIPELINES`.

To obtain more information for a load pipeline, query the pipeline's `OPERATION_ID`:

```
SELECT PIPELINE_NAME, OPERATION_ID FROM USER_CLOUD_PIPELINES
     WHERE PIPELINE_NAME = 'MY_TREE_DATA';

PIPELINE_NAME OPERATION_ID
------------- ------------
MY_TREE_DATA            41
```

Next, query either `USER_LOAD_OPERATIONS` or `DBA_LOAD_OPERATIONS` with a `WHERE` clause predicate on the `ID` column (using the `OPERATION_ID` value).

For example:

```
SELECT ID, TYPE, LOGFILE_TABLE, BADFILE_TABLE, STATUS_TABLE FROM
USER_LOAD_OPERATIONS
```

```
     WHERE ID = 41;

ID TYPE     LOGFILE_TABLE     BADFILE_TABLE     STATUS_TABLE
-- -------- ----------------- ----------------- --------------------
41 PIPELINE PIPELINE$9$41_LOG PIPELINE$9$41_BAD PIPELINE$9$41_STATUS
```

This query shows `ID`, `TYPE`, `LOGFILE_TABLE`, `BADFILE_TABLE` if it exists, and the `STATUS_TABLE`. You can view these tables for additional pipeline load information.

Pipeline Status Table Details

Column	Datatype	Description
`ID`	`NUMBER`	Unique number assigned to the pipeline.
`NAME`	`VARCHAR2(4000)`	Name of the pipeline.
`BYTES`	`NUMBER`	Bytes
`CHECKSUM`	`VARCHAR2(128)`	Checksum
`LAST_MODIFIED`	`TIMESTAMP(6) WITH TIME ZONE`	Last modification time for the pipeline.
`STATUS`	`VARCHAR2(30)`	The `STATUS` value is one of: • `COMPLETED`: The file operation completed successfully. • `FAILED`: The file operation failed, a retry may be attempted two times. • `PENDING`: The file operation has not yet started. • `RUNNING`: The file operation is currently in progress. • `SKIPPED`: The file operation skipped.
`ERROR_CODE`	`NUMBER`	Error code
`ERROR_MESSAGE`	`VARCHAR2(4000)`	Error message
`START_TIME`	`TIMESTAMP(6) WITH TIME ZONE`	Start time for the pipeline.
`END_TIME`	`TIMESTAMP(6) WITH TIME ZONE`	End time for the pipeline.
`SID`	`NUMBER`	The session `SID` and`SERIAL#` indicate the job session that was running the pipeline load operation.
`SERIAL#`	`NUMBER`	The session `SID` and`SERIAL#` indicate the job session that was running the pipeline load operation.
`ROWS_LOADED`	`NUMBER`	Number of rows loaded.
`OPERATION_ID`	`NUMBER`	Reserved for future use.

Pipeline Log File and Bad File Tables

To obtain the log file and bad file names for a load pipeline, query the pipeline's `OPERATION_ID`. For example:

```
SELECT PIPELINE_NAME, OPERATION_ID FROM USER_CLOUD_PIPELINES
     WHERE PIPELINE_NAME = 'MY_TREE_DATA';
```

```
PIPELINE_NAME OPERATION_ID
------------- ------------
MY_TREE_DATA            41
```

Next, query either `USER_LOAD_OPERATIONS` or `DBA_LOAD_OPERATIONS` with a `WHERE` clause predicate on the `ID` column (using the `OPERATION_ID` value).

For example:

```
SELECT ID, TYPE, LOGFILE_TABLE, BADFILE_TABLE, STATUS_TABLE FROM
USER_LOAD_OPERATIONS
     WHERE ID = 41;

ID TYPE     LOGFILE_TABLE     BADFILE_TABLE     STATUS_TABLE
-- -------- ----------------- ----------------- --------------------
41 PIPELINE PIPELINE$9$41_LOG PIPELINE$9$41_BAD PIPELINE$9$41_STATUS
```

This query shows `ID`, `TYPE`, `LOGFILE_TABLE`, `BADFILE_TABLE` if it exists, and the `STATUS_TABLE`. You can view these tables for additional pipeline load information.

View the pipeline log file table to see a complete log of the pipeline's load operations.

For example:

```
SELECT * FROM PIPELINE$9$41_LOG;
```

View the pipeline bad file table to see details on input format records with errors. The bad file table show information for the rows reporting errors during loading. Depending on the errors shown in the log file table and the rows shown in the pipeline's bad file table, you might be able to correct the errors either by modifying the pipeline `format` attribute options, or by modifying the data in the file you are loading.

For example:

```
SELECT * FROM PIPELINE$9$41_BAD;
```

See Monitor and Troubleshoot Loads for more information.

Use Oracle Maintained Pipelines

Autonomous Database provides built-in pipelines. These pipelines are preconfigured and can be started by the ADMIN user. Oracle Maintained pipelines are owned by ADMIN user.

The Oracle Maintained pipelines are:

- `ORA$AUDIT_EXPORT`: This pipeline exports the database audit logs to object store in JSON format and runs every 15 minutes after starting the pipeline (based on the `interval` attribute value).
- `ORA$APEX_ACTIVITY_EXPORT`: This pipeline exports the Oracle APEX workspace activity log to object store in JSON format. This pipeline is preconfigured with the SQL query for retrieving APEX activity records and runs every 15 minutes after starting the pipeline (based on the `interval` attribute value).

To configure and start an Oracle Managed pipeline:

1. Determine the Oracle Managed Pipeline you want to use: `ORA$AUDIT_EXPORT` or `ORA$APEX_ACTIVITY_EXPORT`.
2. Set the `credential_name` and `location` attributes.

   For example:

```
BEGIN
  DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
    pipeline_name => 'ORA$AUDIT_EXPORT',
    attribute_name => 'credential_name',
    attribute_value => 'DEF_CRED_OBJ_STORE'
  );
  DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
    pipeline_name => 'ORA$AUDIT_EXPORT',
    attribute_name => 'location',
    attribute_value => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/'
  );
END;
/
```

   The log data from database is exported to the object store location you specify.

   See SET_ATTRIBUTE Procedure for more information.

   By default the Oracle Maintained Pipelines use `OCI$RESOURCE_PRINCIPAL` as the `credential_name`. See Enable Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.
3. Optionally, set the `interval`, `format`, or `priority` attributes.

   See SET_ATTRIBUTE Procedure for more information.
4. Start the pipeline.

   See START_PIPELINE Procedure for more information.

The Data Load Page

Use the Data Load page to make more data available to your Oracle Autonomous Database. You can load data from files or databases, from links to external databases or cloud storage files, or from a live feed of data from cloud storage.

See The Data Load Page for more information.

- The Data Load Page
  Use the Data Load dashboard page to make more data available to your Oracle Autonomous Database. You can load data from files or databases, from links to external databases or cloud storage files, or from a live feed of data from cloud storage.

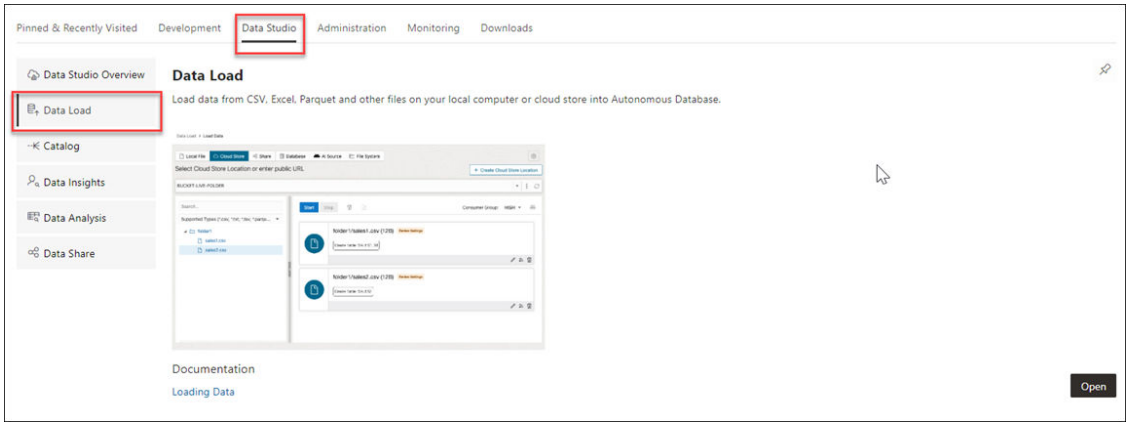
The Data Load Page

Use the Data Load dashboard page to make more data available to your Oracle Autonomous Database. You can load data from files or databases, from links to external databases or cloud storage files, or from a live feed of data from cloud storage.

From the Data Load page, you can also explore the data in your autonomous database and manage your cloud storage locations.

> **Note:**
>
> If you do not see the Data Load card then your database user is missing the required DWROLE role.

To reach the Data Load page, click the **Data Studio** tab in the Database Actions page and select the **Data Load** pane.



You can alternatively click the Selector icon and select **Data Load** from the Data Studio menu in the navigation pane.

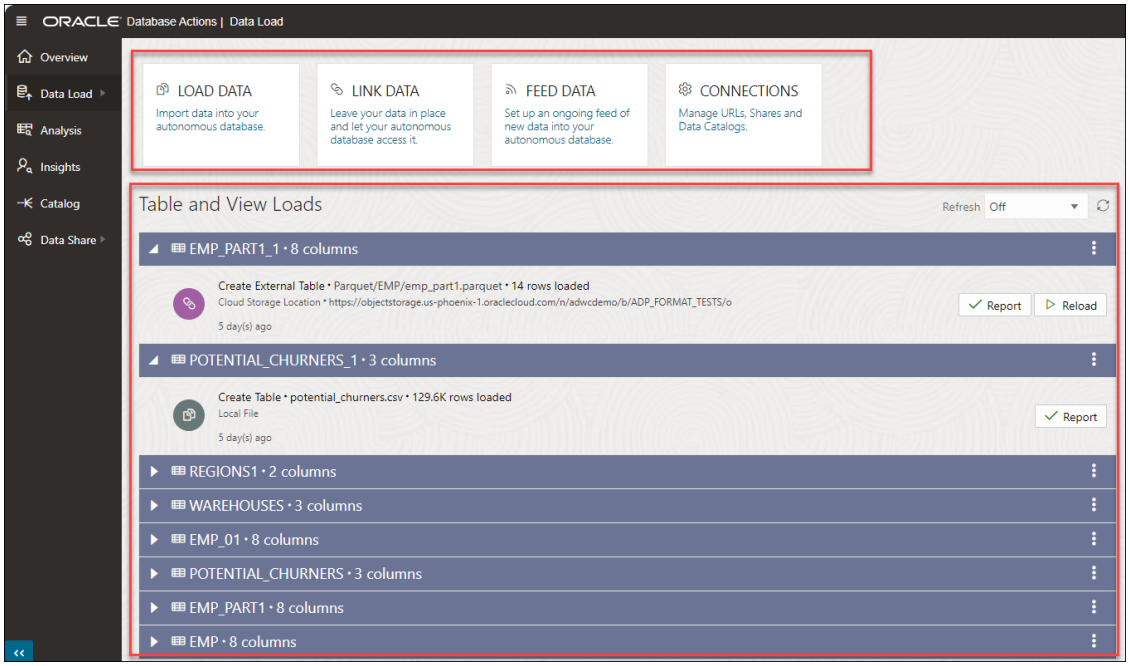
The Data Load Dashboard provides a central location for all data load and live feed tasks. The various icons on the dashboard, provide access to loading data, linking data, setting up an ongoing feed, checking data load jobs, and managing connections.

You can view the following tiles on the top section of the dashboard:

- Load Data
- Link Data
- Feed Data
- Connections

The bottom section of the dashboard displays the tables, views, and life feed you load into the tool.

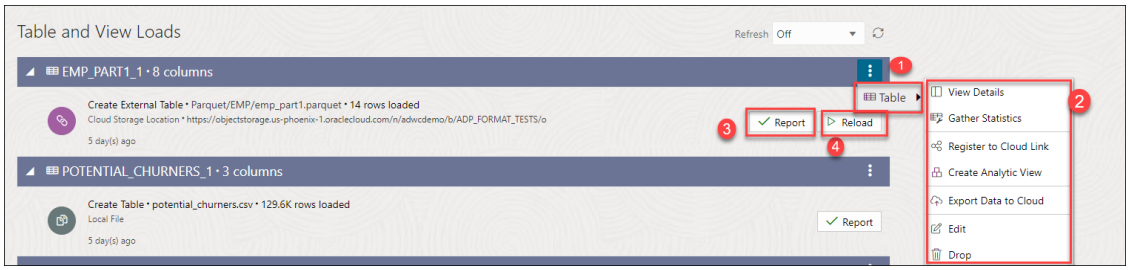
The following image illustrates the **Data Load Dashboard**.



After drilling down onto a table, in the **Table and View Loads** section, you can view the details of the table load such as the name of the table, the number of rows and columns in the table, and the time you load the table.

Tables which are not active for last 7 days are collapsed.

You can also view the source location of the data you load. Click the location to view objects in the location.



Click the **Actions** icon beside the table name and select **Table** to view the following properties of the table:

- View Details
- Gather Statistics
- Register to Cloud Link
- Create Analytic View
- Export Data to Cloud
- Edit
- **Drop** : Select **Drop** to delete the table.

Select **Report** to view a report of total rows processed successfully and failed for the selected table.

Select **Reload** to reload the source of the data load job. Clicking Reload takes you to the **Link Data** page where you can link your data source to the Autonomous Database.

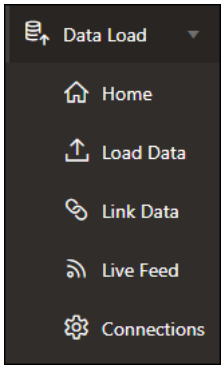
> **Note:**
>
> For tables you load via Cloud Store, you can view **Create Live Feed** icon. See Create Live Feed from Data Load. After the creation of live feed is successful, you will view a Live Feed badge besides the table load header.

In case of errors, you can reload data with suggested fixes.

To load or create links to data or create live table feeds, on the Data Load dashboard, select a combination of an operation and a data source location. The following table lists the operations and the source locations that support those operations.

Operation	Source Location	Description
Load data	Local file Database Cloud storage Directory Share AI Sources	Load data from files on your local device, from remote databases, or from cloud storage into tables in your Oracle Autonomous Database.
Link data	Database Objects in Oracle Cloud Infrastructure (OCI) Catalog	Create external tables or views in your Oracle Autonomous Database that link to data in cloud storage or remote databases. Changes to the source data automatically appear in the target objects.
Feed data	Cloud storage	Set up a feed of data from a cloud storage bucket into a table. Changes to the source data load into the target table as scheduled or on demand.
Connections	Cloud storage Data Catalog links Share Providers	Create a cloud storage location. Subscribe to Share Provider and Register Data Catalog. You can manage cloud storage locations, Data Catalog and Shares.

The left navigation pane of the Data Load Dashboard provides links to the Data Load features and pages. When you select a link from the left navigation pane, the corresponding page appears in the target content area.



The following links make up the Data Load navigation pane:

- Home: This is the current Data Load dashboard that you view.
- Load Data
- Link Data
- Live Feed
- Connections
- Data Studio Preferences
  This topic explains the Data Studio Preferences setting on the **Load Data**, **Link Data** and the **Connections** page under Data Load.
- Managing Connections
  Connections that are established from the Data Studio to the cloud, catalogs and shares are listed on this page.
- Loading Data
  You can load data from files on your local device, from remote databases, or from cloud storage buckets from directories and share providers.
- Linking Data
  You can link to data in remote databases or in cloud storage buckets.
- Feeding Data
  You can run a live table feed on demand, on a schedule, or as the result of a notification.

Data Studio Preferences

This topic explains the Data Studio Preferences setting on the **Load Data**, **Link Data** and the **Connections** page under Data Load.

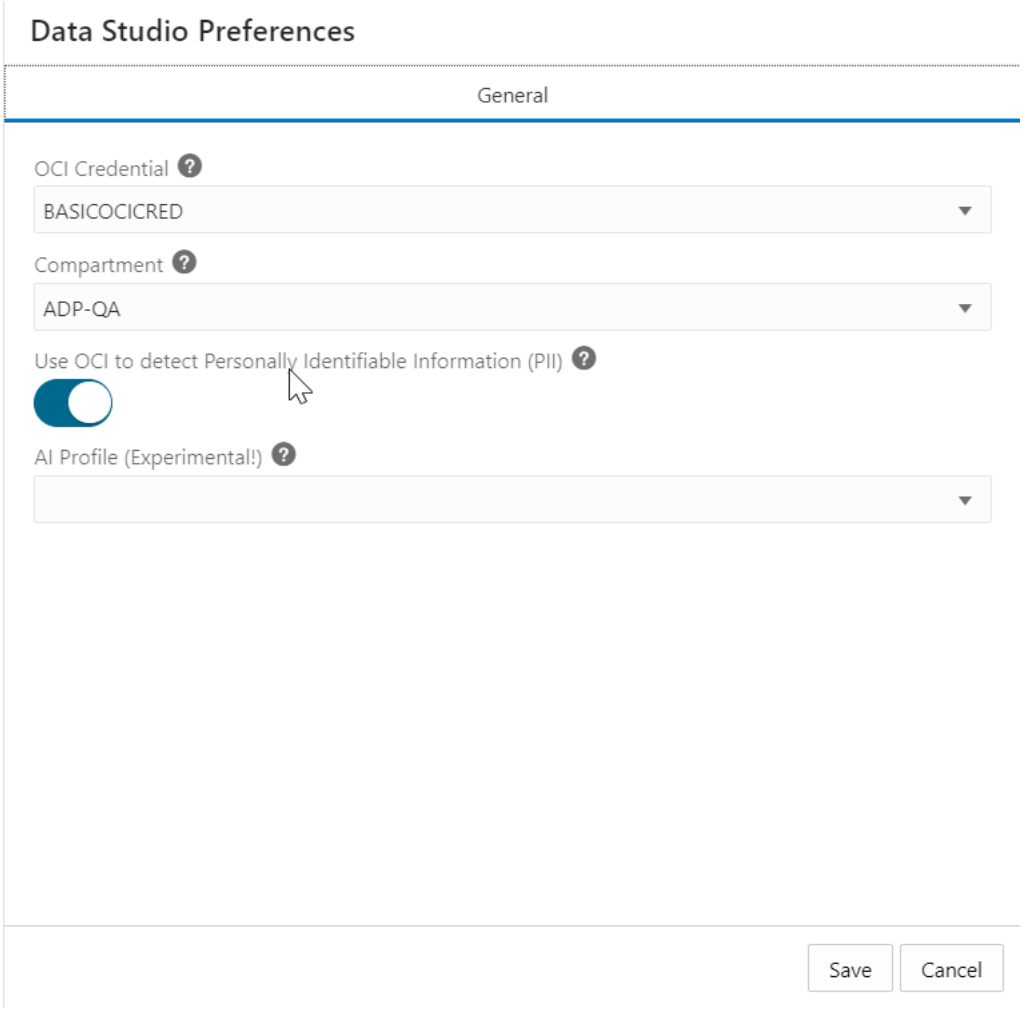
This setting on the **Load Data**, **Link Data** and the **Connections** page enables you to select this data during the load. This setting sets a variety of general preferences and defines the data you can load.

To set the Data Studio Preferences:

1. Select the **Data Studio Preferences** icon

   in the top-right corner of the data load page to view the Data Studio Preferences wizard and access this feature.



2. In the General tab of the Data Studio Preferences screen, select the OCI Credential from the drop-down. See Create Oracle Cloud Infrastructure Native Credentials to refer to the steps you need to follow to establish cloud storage connection from Data Studio to Oracle Cloud Infrastructure (OCI) Object service. You will use this credential to access various OCI services such as OCI languages, Identity, and storage Buckets. Any bucket in the default compartment you set in this wizard automatically appears in the list on the load from cloud storage page. The Data Load tool uses the OCI Credential to list compartments, call the OCI Language PII API, list buckets for data load from cloud Store, and list buckets when creating a Cloud Storage Link. You can access your buckets without explicitly creating a cloud storage link.
3. Select your root compartment from the **Compartment** drop-down which lists the buckets in the Object Storage compartment.
4. Select **Use OCI to detect Personally Identifiable Information (PII)** to warn you about loading private and sensitive data such as your name, date of birth, social security number, and email address.

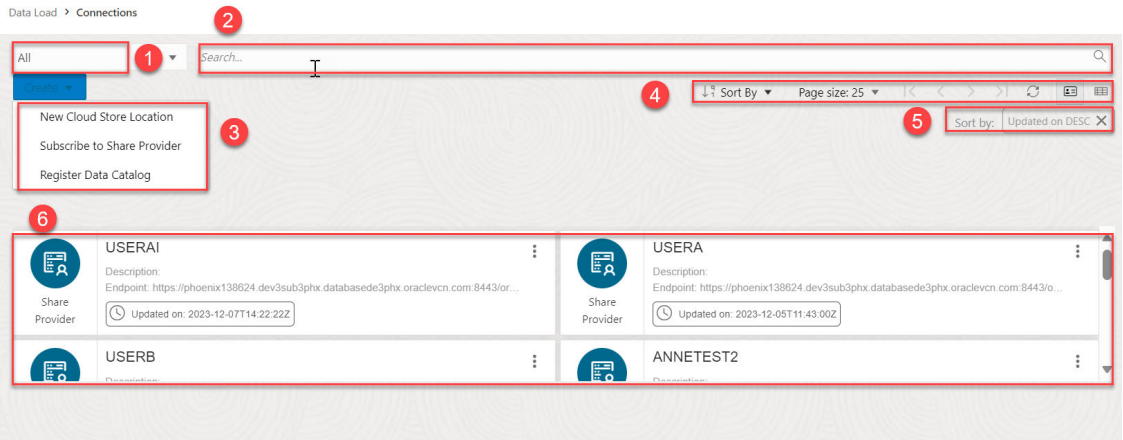
> **Note:**
>
> The Data Preferences feature provides a facility for you to mask sensitive information before it is loaded into the tool. You will receive a warning notification that informs you when you are loading sensitive data into the tool. You can review the data and upload it into the tool.

5. Select **AI Profile (Experimental)** from the drop-down to use this AI profile for facilitating and configuring the translation of natural language prompts to SQL statements. If you have Select AI set up using either OpenAI or Cohere, you can enhance your data by loading small reference data tables that are generated by large language models. Try out the suggested prompts, or come up with your own to generate data to load into the autonomous database.
6. Click **Save** to save the current preferences once all the settings are complete.

Managing Connections

Connections that are established from the Data Studio to the cloud, catalogs and shares are listed on this page.

The Connections page contains the Search for entities such as Cloud Storage Links, Data Catalog Links, and Share Providers, and a list of entity cards. You can enter the entity you are looking for in the field or click one of the entity card from the list. You can register the cloud store you want to use from this page. You can also register data catalogs and subscribe to a share provider.



The Connections page consists of the following:

1. **Entity Selector**: You can Use the drop-down lists to select the entity from which the connection is created. You can select from Cloud Storage Links, Data Catalog Links, Share Provider or all.
2. **Search field**: Searches for entities in the field by name. The search functionality is not case-sensitive, retrieves all matching entries, and does not require the use of wild card characters.
3. **Create drop-down**: The options available in the Create drop-down are:
   - **New Cloud Store Location**: Before you can load data from a cloud store, you must establish a connection to the cloud store you want to use. You can select cloud store location from the cloud store locations field. A cloud storage link is a connection to a

bucket in a cloud store. You can view the existing cloud storage links and add new ones. Refer to the Create Credentials to add a cloud store location.

- **Subscribe to a Share Provider**: Upload the JSON profile file and create a share provider description To subscribe, you need to use the information contained in the uploaded JSON profile you received from the share provider. From the Consume Share feature of the Data Share tool, you upload the JSON profile and follow the subscribe wizard. Refer to the Consume Versioned Share for more details.
- **Register Data Catalogs**: You can create a connection by registering a data source as a data asset in your data catalog. You can view, delete and rename the data catalogs. Refer to the Register Data Catalog chapter.

4. The toolbar consists of the following buttons:
   - **Sort By**

     To select sorting values, click the **Sort By** button to open the list of options. Then click the **Ascending** or **Descending** icon next to one or more of the sorting values. For example, if you select the **Ascending** icon next to **Entity name** and the **Descending** icon next to **Entity type**, the entities will be sorted in alphabetical order by entity name and then in reverse alphabetical order by entity type.

     Click **Reset** in the list to clear the choices in the list.
   - **Page size**

     By default, up to 25 entities are displayed on the page. If you want more entities on a page, select a number from this list.
   - **Previous** and **Next**

     If the search results are displayed on multiple pages, click these buttons to navigate through the pages.
   - **Refresh**

     Click to refresh the data load jobs shown on the page, based on the current search field.
5. The sorting values you choose are listed next to the **Sort by** label beneath the toolbar. Click the **X** icon on a sorting value to remove it.
6. Display area: The area below the Create drop-down field displays the entity carts returned by a search.

View Entity Details

- To view details about the existing entities, click **Actions**.
- For a Share Provider, refer to the View Share Provider Entity detail.
- For a Cloud Storage Link, click **Actions** to perform the following operations:
  - Select **View Details** to view details about the table.
  - Select **Objects** to view objects available in the selected storage link. You can click a file on the navigator pane to view it on the display area.
  - Selecting **Link Tables** opens the Link Data page on the Data Load tool with the selected cloud storage link on the Cloud Location URL field. You can link data present in the cloud storage to the Autonomous Database. See Linking to Objects in Cloud Storage.
  - Selecting **Load Tables** opens the Load Data page on the Data Load tool with the selected cloud storage link on the Cloud Location URL field. You can load data present

in the cloud storage to the Autonomous Database. See Loading Data from Cloud Storage.

  - Selecting **Create Live Table Feed** opens the Create Live Table feed wizard with he selected cloud storage link on the Cloud Location URL field. See Feeding Data to view more details.
  - Select **Edit** to update any details on the cloud storage location. See Create Credentials to view details on creation of cloud storage location.
  - Select **Rename** to rename the cloud store location to a different name.
  - Select **Delete** to delete the cloud store location.

- Create Credentials
  The procedure for creating a credential varies depending on the cloud storage provider. If your source files reside in a cloud store provided by one of the following, see the example for that provider.
- Create Oracle Cloud Infrastructure Native Credentials
  To establish cloud storage connection from Data Studio to Oracle Cloud Infrastructure (OCI) Object storage service, you must configure the cloud storage location with your OCI authentication details. You can create Oracle Cloud Infrastructure (OCI) Native Credentials by using the CREATE_CREDENTIAL procedure of DBMS_CLOUD package.
- Create Amazon Web Services (AWS) Credentials
  To access AWS Glue Catalog you must have AWS credentials that you can create via the Create Credential wizard. This wizard stores cloud service credentials in the Autonomous Database.
- Register Data Catalog
  You can register data catalogs you want to use with registering the data catalog.
- Manage Resource Principal with DBMS_CLOUD
  The Oracle Cloud Infrastructure (OCI) Resource Principal allows applications and services running within the OCI to access an Autonomous Database without the need for traditional database credentials (e.g., username and password).

Create Credentials

The procedure for creating a credential varies depending on the cloud storage provider. If your source files reside in a cloud store provided by one of the following, see the example for that provider.

- **Oracle Cloud Infrastructure (OCI)**, see Create an OCI Cloud Store Location.
- **Amazon S3**, or you are calling an AWS API, see Create an Amazon S3 Cloud Store Location.
- **Microsoft Azure Blob Storage** or you are calling an Azure API, see Create a Microsoft Azure Cloud Store Location.
- **Google Cloud Storage**, see Create a Google Cloud Storage Location.
- **Other (Swift compatible)** cloud storage, see Create an Other (Swift Compatible) Cloud Store Location.
- **OCI cloud storage** by using native OCI credentials, see the subsequent section.

> **Note:**
>
> This method is suggested if you need to use the OCI REST APIs. You need to use the Native OCI credentials if you want to use the OCI REST APIs. To create an OCI cloud storage location using Oracle Cloud Infrastructure Signing Keys, you must first Create Oracle Cloud Infrastructure Native Credentials.

Create an OCI Cloud Store Location

1. On the **Connections** page, click **Create** and select **Create New Cloud Store Location**. This opens the Add Cloud Store Location wizard.
2. In the Storage Settings tab of the Add Cloud Store Location box, enter a name for the cloud storage link. For example:

   ```
   My_Cloud_Store
   ```

3. (Optional) In the **Description** field, enter a description for the link. For example:

   ```
   My cloud storage link.
   ```

4. Click **Select Credential** and **Create Credential** to create a credential. This opens a Create Credential wizard.
   In the Credentials section, select **Cloud Username and Password**.
5. From the **Cloud Store** drop-down list, select **Oracle**.
6. Enter a name in the **Credential Name** field. The name must conform to Oracle object naming conventions, which do not allow spaces or hyphens. For example:

   ```
   my_credential
   ```

7. For an OCI cloud store, in the **Oracle Cloud Infrastructure User Name** field, enter your OCI user name. You must use Oracle Cloud Infrastructure User Name from your profile in the OCI console. For example:

   ```
   oracleidentitycloudservice/foo@example.com
   ```

   or

   ```
   default/foo@example.com
   ```

8. For an OCI cloud store, in the **Auth Token** field, enter your auth token. For example:

   ```
   LPB>Ktk(1M1SD+a]+r
   ```

9. In the **Bucket URI** field, enter the URI and bucket for your OCI instance bucket.
   a. To get the URI and bucket, go to the bucket in the Object Storage compartment in your Oracle Cloud Instance.
   b. In the Objects group, click the Actions (three vertical dots) icon for a file in the bucket, then click **View Object Details**.
   c. Copy all of the URL Path (URI) except for the file name. Be sure to include the trailing slash. For example, for the file `https://objectstorage.us-`

`phoenix-1.oraclecloud.com/n/myoci/b/my_bucket/o/MyFile.csv`, select the following:

```
 https://objectstorage.us-phoenix-1.oraclecloud.com/n/myoci/b/
my_bucket/o/
```

d. Paste the string into the **URI + Bucket** field.

10. Click **Next**.
The dialog box progresses to the Cloud Data tab. This tab lists the objects available on this cloud storage location in the display area.

> **Note:**
>
> The display area is blank when we create a new cloud storage location.

11. Click **Create** create the cloud storage location.

Create an Amazon S3 Cloud Store Location

1. On the **Connections** page, click **Create** and select **Create New Cloud Store Location**. This opens the Add Cloud Store Location wizard.
2. In the Storage Settings tab of the Add Cloud Store Location box, enter a name for the cloud storage link. For example:

```
My_Cloud_Store
```

3. (Optional) In the **Description** field, enter a description for the link. For example:

```
My cloud storage link.
```

4. Click **Select Credential** and **Create Credential** to create a credential. This opens a Create Credential wizard.
In the Credentials section, select **Cloud Username and Password**.
5. From the **Cloud Store** drop-down list, select **Oracle**.
6. Enter a name in the **Credential Name** field. The name must conform to Oracle object naming conventions, which do not allow spaces or hyphens. For example:

```
my_credential
```

7. From the **Cloud Store** drop-down list, select **Amazon S3**.
8. Enter a name in the **Credential Name** field. The name must conform to Oracle object naming conventions, which do not allow spaces or hyphens. For example:

```
my_credential
```

9. In the **AWS Access Key ID** field, enter your AWS access key ID. For example:myAccessKeyID
10. In the **AWS Secret Access Key** field, enter your AWS secret access key. For information on AWS access keys, see Managing access keys for IAM users.

11. In the **Bucket URI** field, enter the URI and bucket for your Amazon S3 bucket. For example:

    ```
    https://s3.us-west-2.amazonaws.com/my_bucket
    ```

12. Click **Next**.
    The dialog box progresses to the Cloud Data tab. This tab lists the objects available on this cloud storage location in the display area.

    > **Note:**
    >
    > The display area is blank when we create a new cloud storage location.

13. Click **Create** create the cloud storage location.

Create an Microsoft Azure Cloud Store Location

1. On the **Connections** page, click **Create** and select **Create New Cloud Store Location**. This opens the Add Cloud Store Location wizard.
2. In the Storage Settings tab of the Add Cloud Store Location box, enter a name for the cloud storage link. For example:

    ```
    My_Cloud_Store
    ```

3. (Optional) In the **Description** field, enter a description for the link. For example:

    ```
    My cloud storage link.
    ```

4. Click **Select Credential** and **Create Credential** to create a credential. This opens a Create Credential wizard.
    In the Credentials section, select **Cloud Username and Password**.
5. From the Cloud Store drop-down list, select **Microsoft Azure**.
6. Enter a name in the **Credential Name** field. The name must conform to Oracle object naming conventions, which do not allow spaces or hyphens. For example:

    ```
    my_credential
    ```

7. In the **Azure Storage Account Name** field, enter the name of your Azure storage account. For example:

    ```
    myaccount
    ```

8. In the **Azure Storage Account Access Key** field, enter your Azure access key.
    For information on Azure storage accounts, see Create a storage account.
9. In the **Bucket URI** field, enter the URI and bucket for your Microsoft Azure bucket. For example:

    ```
    https://myaccount.blob.core.windows.net/mycontainer
    ```

10. Click **Next**.
    The dialog box progresses to the Cloud Data tab. This tab lists the objects available on this cloud storage location in the display area.

> **Note:**
>
> The display area is blank when we create a new cloud storage location.

11. Click **Create** to create the cloud storage location.

Create an Microsoft Azure Cloud Store Location

1. On the **Connections** page, click **Create** and select **Create New Cloud Store Location**. This opens the Add Cloud Store Location wizard.
2. In the Storage Settings tab of the Add Cloud Store Location box, enter a name for the cloud storage link. For example:

   ```
   My_Cloud_Store
   ```

3. (Optional) In the **Description** field, enter a description for the link. For example:

   ```
   My cloud storage link.
   ```

4. Click **Select Credential** and **Create Credential** to create a credential. This opens a Create Credential wizard.
   In the Credentials section, select **Cloud Username and Password**.
5. From the Cloud Store drop-down list, select **Microsoft Azure**.
6. Enter a name in the **Credential Name** field. The name must conform to Oracle object naming conventions, which do not allow spaces or hyphens. For example:

   ```
   my_credential
   ```

7. In the **Azure Storage Account Name** field, enter the name of your Azure storage account. For example:

   ```
   myaccount
   ```

8. In the **Azure Storage Account Access Key** field, enter your Azure access key.
   For information on Azure storage accounts, see Create a storage account.
9. In the **Bucket URI** field, enter the URI and bucket for your Microsoft Azure bucket. For example:

   ```
   https://myaccount.blob.core.windows.net/mycontainer
   ```

10. Click **Next**.
    The dialog box progresses to the Cloud Data tab. This tab lists the objects available on this cloud storage location in the display area.

> **Note:**
>
> The display area is blank when we create a new cloud storage location.

11. Click **Create** to create the cloud storage location.

Create a Google Cloud Store Location

1. On the **Connections** page, click **Create** and select **Create New Cloud Store Location**. This opens the Add Cloud Store Location wizard.
2. In the Storage Settings tab of the Add Cloud Store Location box, enter a name for the cloud storage link. For example:

   ```
   My_Cloud_Store
   ```

3. (Optional) In the **Description** field, enter a description for the link. For example:

   ```
   My cloud storage link.
   ```

4. Click **Select Credential** and **Create Credential** to create a credential. This opens a Create Credential wizard.
   In the Credentials section, select **Cloud Username and Password**.
5. Enter a name in the **Credential Name** field. The name must conform to Oracle object naming conventions, which do not allow spaces or hyphens. For example:

   ```
   my_credential
   ```

6. From the **Cloud Store** drop-down list, select **Google**.
7. In the **HMAC Access Key** field, enter your HMAC access ID. For example:

   ```
   GOOGTS1C3LPB3KTKSDMB2BFD
   ```

8. In the **HMAC Access Secret** field, enter your HMAC secret. For information on HMAC keys, see HMAC Keys.
9. In the Storage Settings tab of the Add Cloud Store dialog box, enter a name for the cloud storage link. For example:

   ```
   My_Cloud_Store
   ```

10. (Optional) In the **Description** field, enter a description for the link. For example:

    ```
    My cloud storage link.
    ```

11. In the **Bucket URI** field, enter the bucket and URI for your Google bucket. For example:

    ```
    https://my_bucket.storage.googleapis.com
    ```

12. Click **Next**.
    The dialog box progresses to the Cloud Data tab. This tab lists the objects available on this cloud storage location in the display area.

    > **Note:**
    >
    > The display area is blank when we create a new cloud storage location.

13. Click **Create** to create the cloud storage location.

Create an Other (Swift Compatible) Cloud Store Location

1. On the **Connections** page, click **Create** and select **Create New Cloud Store Location**. This opens the Add Cloud Store Location wizard.
2. In the Storage Settings tab of the Add Cloud Store Location box, enter a name for the cloud storage link. For example:

   ```
   My_Cloud_Store
   ```

3. (Optional) In the **Description** field, enter a description for the link. For example:

   ```
   My cloud storage link.
   ```

4. Click **Select Credential** and **Create Credential** to create a credential. This opens a Create Credential wizard.
   In the Credentials section, select **Cloud Username and Password**.

   > **Note:**
   >
   > If you have the user OCID, tenancy OCID, private key and fingerprint, select **Oracle Cloud Infrastructure Signing Keys** and refer to the *Create an OCI Cloud storage location using Oracle Cloud Infrastructure Signing Keys* section of this topic.

5. From the Cloud Store drop-down list, select **Other (Swift Compatible)**.
6. Enter a name in the **Credential Name** field. The name must conform to Oracle object naming conventions, which do not allow spaces or hyphens. For example:

   ```
   my_credential
   ```

7. In the **Access User Name** field, enter your access user name. For example:

   ```
   OTHER_KEY123...
   ```

8. In the **Access Key** field, enter your access key.
9. In the **Bucket URI** field, enter the URI and bucket for your cloud store bucket. For example:

   ```
   https://someswiftcompatibleprovider.com/my_bucket
   ```

10. Click **Next**.The dialog box progresses to the Cloud Data tab. This tab lists the objects available on this cloud storage location in the display area. The display area is blank when we create a new cloud storage location.
11. Click **Create** to create the cloud storage location.

Create an OCI Cloud storage location using Oracle Cloud Infrastructure Signing Keys

To create an OCI Cloud storage location using Oracle Cloud Infrastructure Signing Keys:

1. On the **Connections** page, click **Create** and select **Create New Cloud Store Location**. This opens the Add Cloud Store Location wizard.

2. In the Storage Settings tab of the Add Cloud Store Location box, enter a name for the cloud storage link. For example:

   ```
   My_Cloud_Store
   ```

3. (Optional) In the **Description** field, enter a description for the link. For example:

   ```
   My cloud storage link.
   ```

4. Click **Select Credential** and **Create Credential** to create a credential. This opens a Create Credential wizard.
   If you have the user OCID, tenancy OCID, private key and fingerprint, select **Oracle Cloud Infrastructure Signing Keys**.

   > **Note:**
   >
   > If you only have a username and password select **Cloud Username and Password** in this step and refer to the *Create an Other (Swift Compatible) Cloud Store Location* section of this topic.

5. Specify the following information about your OCI account:

   Credential Name: Specify a name to identify the credentials. See Create an OCI Credential Object to enter the Credential name to specify the credential name.

6. **Fingerprint:** The fingerprint of the RSA key pair. See Create an OCI Credential Object to enter the Credential name to enter the fingerprint.

7. **Private Key:** The unencrypted private key in the RSA key pair. This should not be encrypted by using any passphrase. See Create an OCI Credential Object to enter the Credential name to enter the private key.

8. **Oracle Cloud Infrastructure Tenancy:** The OCID of the tenant. See Where to Get the Tenancy's OCID and User's OCID for details on obtaining the Tenancy's OCID.

9. **Oracle Cloud Infrastructure User Name:** The OCID of the user. See Where to Get the Tenancy's OCID and User's OCID for details on obtaining the User's OCID.

10. Select **Create Credential**.

11. n the **Bucket URI** field, enter the URI and bucket for your cloud store bucket. For example:

    ```
    https://objectstorage.<region>.oraclecloud.com/<namespace>/b/<bucket>/.
    ```

12. Click **Next**.

13. The dialog box progresses to the Cloud Data tab. This tab lists the objects available on this cloud storage location in the display area.

    > **Note:**
    >
    > The display area is blank when we create a new cloud storage location.

14. Click **Create**.

You will receive a notification that the cloud storage location is created successfully.

Create Oracle Cloud Infrastructure Native Credentials

To establish cloud storage connection from Data Studio to Oracle Cloud Infrastructure (OCI) Object storage service, you must configure the cloud storage location with your OCI authentication details. You can create Oracle Cloud Infrastructure (OCI) Native Credentials by using the CREATE_CREDENTIAL procedure of DBMS_CLOUD package.

Create an Oracle Cloud Infrastructure (OCI) Credential Object

To access Object Storage, you must have credentials that you can create via the CREATE_CREDENTIAL procedure of DBMS_CLOUD package. DBMS_CLOUD supports creation of credential objects that contains OCI native authentication. The DBMS_CLOUD procedure stores cloud service credentials in Autonomous Database.

The DBMS_CLOUD.CREATE_CREDENTIAL procedure is overloaded with the Oracle Cloud Infrastructure-related parameters, including: user_ocid, tenancy_ocid, private_key, and fingerprint. This is for using Oracle Cloud Infrastructure Signing Keys authentication.

Let us create an OCI native authentication credential when creating an object store credential object. In the OCI native authentication, the DBMS_CLOUD.CREATE_CREDENTIAL procedure includes the following parameters:

Table 3-2 DBMS_CLOUD.CREATE_CREDENTIAL parameter descriptions

Parameter	Description
`credential_name`	The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
`user_ocid`	Specifies the user's OCID. See Where to Get the Tenancy's OCID and User's OCID for details on obtaining the User's OCID.
`tenancy_ocid`	Specifies the tenancy's OCID. See Where to Get the Tenancy's OCID and User's OCID for details on obtaining the Tenancy's OCID.
`private_key`	Specifies the generated private key. Private keys generated with a passphrase are not supported. You need to generate the private key without a passphrase. See How to Generate an API Signing Key for details on generating a key pair in PEM format.
`fingerprint`	Specifies a fingerprint. After a generated public key is uploaded to the user's account the fingerprint is displayed in the console. Use the displayed fingerprint for this argument. See How to Get the Key's Fingerprint and How to Generate an API Signing Key for more details.

Here is the syntax of the DBMS_CLOUD.CREATE_CREDENTIAL procedure:

```
DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name IN VARCHAR2,
    user_ocid       IN VARCHAR2,
    tenancy_ocid    IN VARCHAR2,
    private_key     IN VARCHAR2,
    fingerprint     IN VARCHAR2);
```

Once you obtain all the necessary inputs and generate your private key, here is a sample of your CREATE_CREDENTIAL procedure:

```
BEGIN  DBMS_CLOUD.CREATE_CREDENTIAL (    credential_name =>
      'OCI_NATIVE_CRED',    user_ocid             =>

'ocid1.user.oc1..aaaaaaaatfn77fe3fxux3o5lego7glqjejrzjsqsrs64f4jsjrhbsk5qzndq'
,    tenancy_ocid          =>

'ocid1.tenancy.oc1..aaaaaaaapwkfqz3upqklvmelbm3j77nn3y7uqmlsod75rea5zmtmbl574v
e6a',    private_key           =>

'MIIEogIBAAKCAQEAsbNPOYEkxM5h0DF+qXmie6ddo95BhlSMSIxRRSO1JEMPeSta0C7WEg7g8SOSz
hIroCkgOqDzkcyXnk4BlOdn5Wm/BYpdAtTXk0sln2DH/GCH7l9P8xC9cvFtacXkQPMAXIBDv/
zwG1kZQ7Hvl7Vet2UwwuhCsesFgZZrAHkv4cqqE3uF5p/
qHfzZHoevdq4EAV6dZK4Iv9upACgQH5zf9IvGt2PgQnuEFrOm0ctzW0v9JVRjKnaAYgAbqa23j8tKa
pgPuREkfSZv2UMgF7Z7ojYMJEuzGseNULsXn6N8qcvr4fhuKtOD4t6vbIonMPIm7Z/
a6tPaISUFv5ASYzYEUwIDAQABAoIBACaHnIv5ZoGNxkOgF7ijeQmatoELdeWse2ZXll+JaINeTwKU1
fIB1cTAmSFv9yrbYb4ubKCJuYZJeC6I92rT6gEiNpr670Pn5n43cwblszcTryWOYQVxAcLkejbPA7j
Zd6CW5xm/vEgRv5qgADVCzDCzrij0t1Fghicc+EJ4BFvOetnzEuSidnFoO7K3tHGbPgA+DPN5qrO/
8NmrBebqezGkOuOVkOA64mp467DQUhpAvsy23RjBQ9iTuRktDB4g9cOdOVFouTZTnevN6JmDxufu9L
ov2yvVMkUC2YKd+RrTAE8cvRrn1A7XKkH+323hNC59726jT57JvZ+ricRixSECgYEA508e/
alxHUIAU9J/uq98nJY/
6+GpI9OCZDkEdBexNpKeDq2dfAo9pEjFKYjH8ERj9quA7vhHEwFL33wk2D24XdZl6vq0tZADNSzOtT
rtSqHykvzcnc7nXv2fBWAPIN59s9/oEKIOdkMis9fps1mFPFiN8ro4ydUWuR7B2nM2FWkCgYEAxKs/
zOIbrzVLhEVgSH2NJVjQs24S8W+99uLQK2Y06R59L0Sa90QHNCDjB1MaKLanAahP30l0am0SB450kE
iUD6BtuNHH8EIxGL4vX/SYeE/
AF6tw3DqcOYbLPpN4CxIITF0PLCRoHKxARMZLCJBTMGpxdmTNGyQAPWXNSrYEKFsCgYBp0sHr7TxJ1
WtO7gvvvd91yCugYBJAyMBr18YY0soJnJRhRL67A/
hlk8FYGjLW0oMlVBtduQrTQBGVQjedEsepbrAcC+zm7+b3yfMb6MStE2BmLPdF32XtCH1bOTJSqFe8
FmEWUv3ozxguTUam/
fq9vAndFaNre2i08sRfi7wfmQKBgBrzcNHN5odTIV8l9rTYZ8BHdIoyOmxVqM2tdWONJREROYyBtU7
PRsFxBEubqskLhsVmYFO0CD0RZ1gbwIOJPqkJjh+2t9SH7Zx7a5iV7QZJS5WeFLMUEv+YbYAjnXK+d
OnPQtkhOblQwCEY3Hsblj7Xz7o=',    fingerprint           =>
      '4f:0c:d6:b7:f2:43:3c:08:df:62:e3:b2:27:2e:3c:7a');END;/
    PL/SQL procedure successfully completed.
```

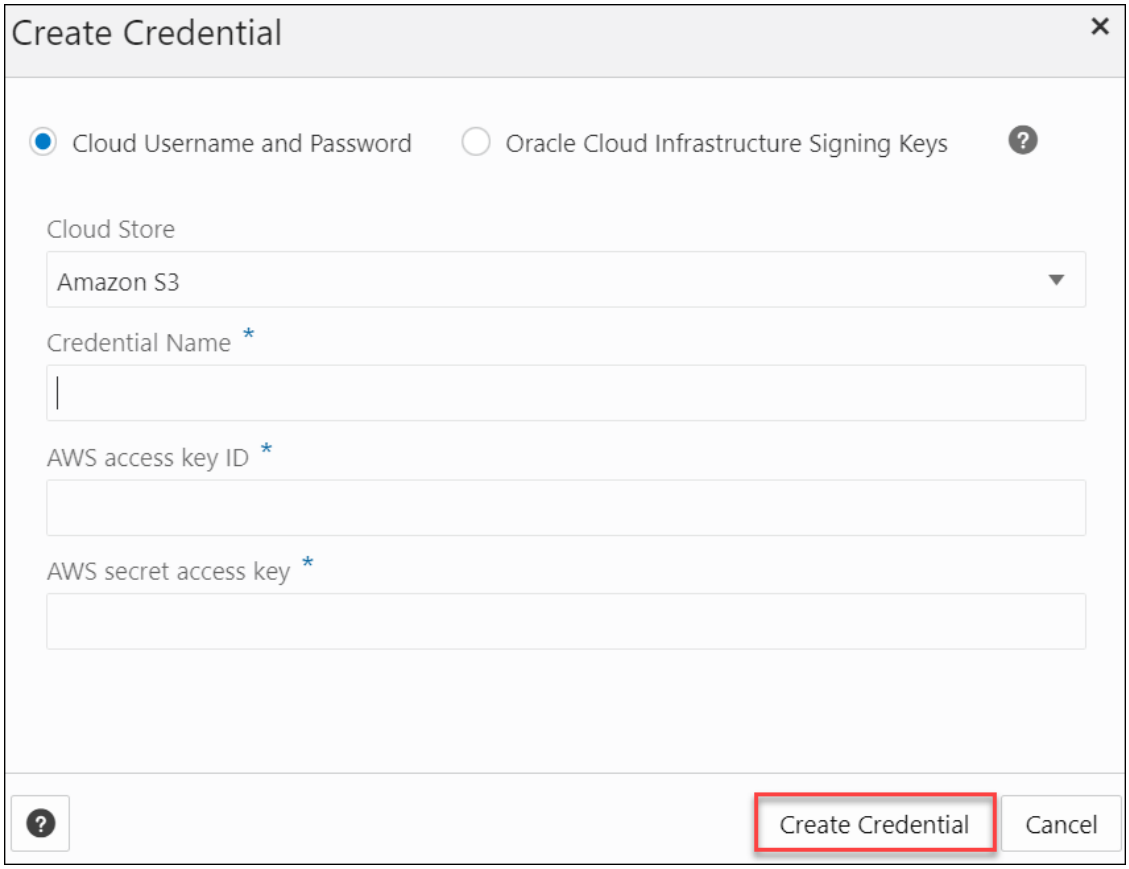
You can now retrieve the new credentials with the following query:

```
SELECT owner, credential_name FROM dba_credentials WHERE credential_name LIKE
'%NATIVE%';  OWNER CREDENTIAL_NAME----- ---------------
    ADMIN OCI_NATIVE_CRED
```

Create Amazon Web Services (AWS) Credentials

To access AWS Glue Catalog you must have AWS credentials that you can create via the Create Credential wizard. This wizard stores cloud service credentials in the Autonomous Database.

Use the `CREATE CREDENTIAL` wizard depending on where your source files reside. To create AWS credentials:



1. Select Cloud Username and Password.
2. Select **Amazon S3** from the Cloud Store drop-down.
3. Enter a name in the Credential Name field. The name must conform to Oracle object naming conventions, which do not allow spaces or hyphens. For example, `MY_AWS_CRED`.
4. In the **AWS Access Key ID** field, enter your AWS access key ID. For example: `myAccessKeyID`, see Managing access keys for IAM users.
5. In the **AWS Secret Access Key** field, enter your AWS secret access key. For information on AWS access keys, see Managing access keys for IAM users.
6. Click **Create Credential** to create AWS credentials.

Register Data Catalog

You can register data catalogs you want to use with registering the data catalog.

Register OCI Data Catalog

To register a data catalog you need to specify the details of the credentials you want to register your data source. A credential object manages a data catalog instance. In OCI native authentication, the DBMS_CLOUD.CREATE_CREDENTIAL procedure includes these parameters: `credential_name`, `user_ocid`, `tenancy_ocid`, `private_key`, and `fingerprint`.

See REF DBMS_DCAT Package to refer to the procedures to add custom parameters such as region and data catalog ID to the Data Catalog. The Data Catalog ID is a unique Oracle cloud Identifier for data catalog instance and region is the data catalog region.

To register Data catalog:

1. From Connections page, click **Create** and select **Register Data Catalog**. This opens Register Data Catalog wizard.
2. In the Catalog Settings tab, specify the following details:
   - **Catalog Name:** `Test`. Enter a name of your choice.
   - **Description:** Specify a description. This is an optional field.
   - Under Data Catalog details, fill in the following field values:
     - **Credential for Data Catalog Connection:** Select a value from the drop-down. The drop-down lists the values of the credentials you create. If you do not have credentials, you can create one. Refer to Create Credentials for more information.
     - **Region:** `us-ashburn-1`. Enter the region name you use while you create the credentials using the DBMS_DCAT package.
     - **Data Catalog ID:** `ocid1.datacatalog.oc1.iad.amaaaaaa7ratcziayxh7uzll24cp3uwzsugfj7qlubak77toiehidpsqsygq`. Enter the data catalog ID from the DBMS_DCAT package.
     - Select the **Register Data Catalog Connection** button to register the data catalog within the autonomous database. You can view this option when you select **OCI Data Catalog** from Catalog Type.
     - Select the **Use separate credential** for object Storage to select the database credentials from the drop-down. You can view this option when you select **OCI Data Catalog** from Catalog Type.
3. Click **Next** to progress to the Register Assets tab. This tab creates a connection with the source Data Catalog objects you select from the list of objects.
4. After successful registering of the data catalog objects, click **Create**.
5. You will receive a notification that the data catalog is created successfully.
   After successful creation of the data catalog, you can view the data catalog entity in the list of entities in the **Connections** page.

Register AWS Glue Catalog

You can integrate Oracle Data Catalog with Amazon's Glue data catalog.

To access the AWS Glue Catalog, register the catalog on the **Connections** page.

This enables you to synchronize Data Catalog metadata with AWS Glue and query data stored in S3 from an Autonomous Database without manually deriving the schema for the external data sources and creating external tables.

Refer to Query External Data with Glue Data Catalog for more details.

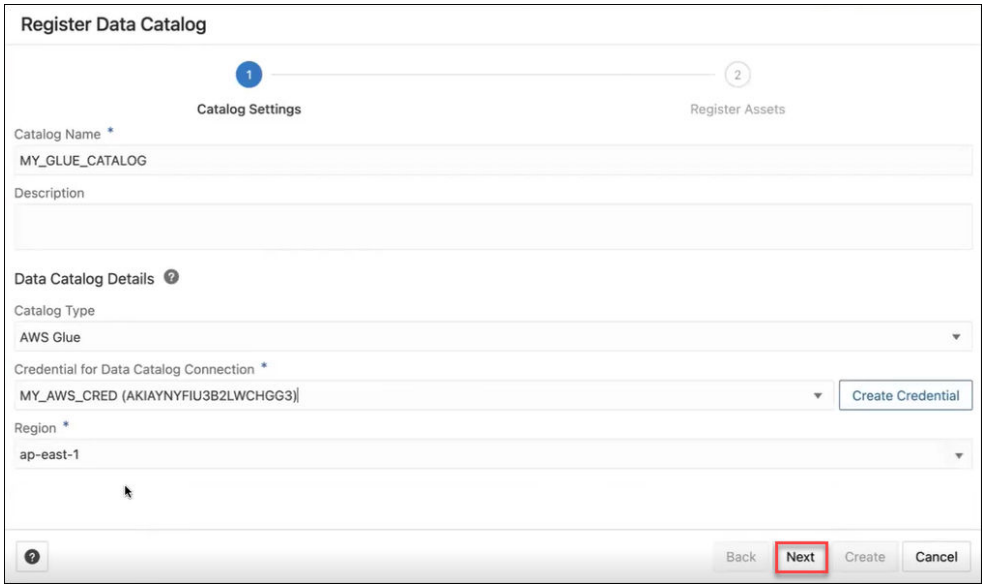
You also need an AWS Credential to be associated with the AWS Glue Catalog. The tool initiates the connection by specifying the following field values:

- Cloud Store
- Credential Name
- AWS access key ID
- AWS Secret Access Key

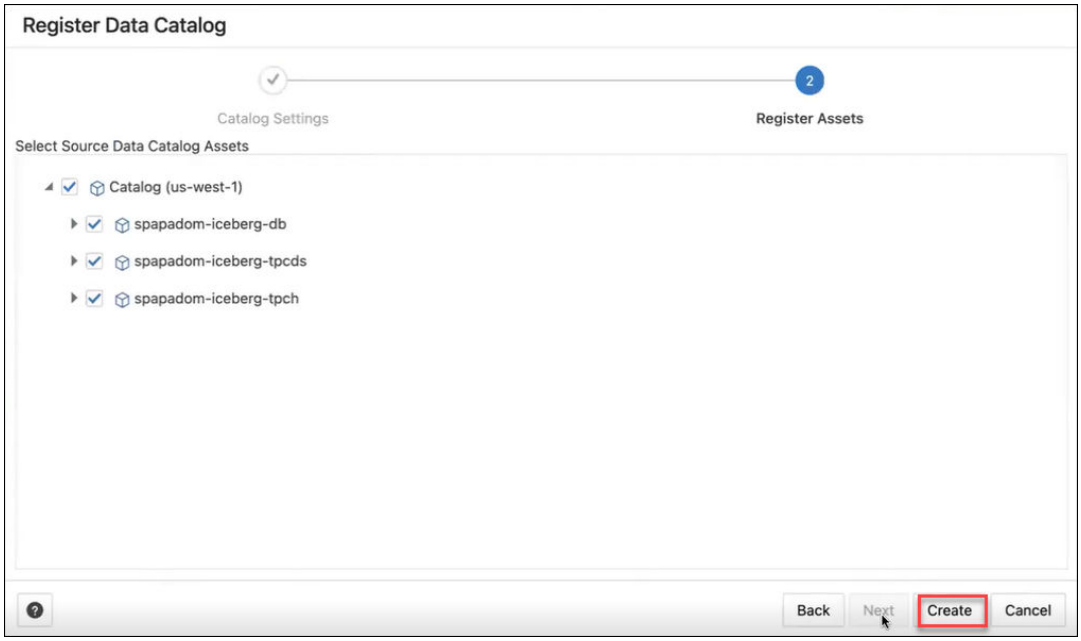
To register the Data Catalog:

1. From the Connections page, click **Create** and select **Register Data Catalog**. This opens the Register Data Catalog wizard.

2. In the Catalog Settings tab, specify the following details:
   - **Catalog Name**: `MY_GLUE_CATALOG`
   - **Description**: Specify a description. This is an optional field.
   - Under Data Catalog Details, fill in the following field values:
     - **Catalog Type**: Select AWS Catalog from the drop-down.
     - **Credential for Data Catalog Connection**: Select a credential value from the drop down. The drop-down lists the values of the credentials you create. If you do not have credentials, you can create one. Refer to Create Amazon Web Services (AWS) Credentials for more information.
     - **Region**: `us-east-1`. Enter the region name you use while you create the AWS credentials.



3. Click **Next** to progress to the Register Assets tab. The Register Assets tab creates a connection with the source data catalog objects you select from the list of objects.
4. After successfully registering the data catalog objects, click **Create**.



5. You will receive a notification that the data catalog has been created successfully. After the successful creation of the data catalog, you can view the data catalog entity in the list of entities on the **Connections** page.

Manage Resource Principal with DBMS_CLOUD

The Oracle Cloud Infrastructure (OCI) Resource Principal allows applications and services running within the OCI to access an Autonomous Database without the need for traditional database credentials (e.g., username and password).

You do not need to create a credential object when you use a resource principal. The Autonomous Database creates and secures the resource principal credentials you use to access the specified Oracle Cloud Infrastructure resources. A resource principal consists of a temporary session token and secure credentials that enable the database to authenticate itself to other Oracle Cloud Infrastructure services.

Prerequisites to Use Resource Principal with Autonomous Database:

- There are several steps required to set up a resource principal on an Autonomous Database: You must create and define Oracle Cloud Infrastructure Identity and Access Management (IAM) policies. See Perform Prerequisites to Use Resource Principal with Autonomous Database for more information.
- Enable the resource principal for the ADMIN user, and optionally enable the resource principal for a database user. See Enable Resource Principal to access OCI Resources.

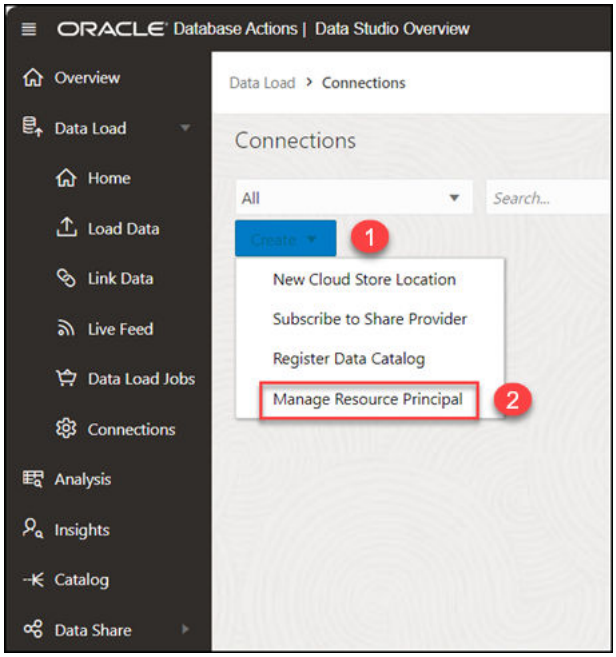
When you authenticate using a resource principal, you do not need to create and manage credentials to access OCI resources. The Autonomous Database makes the resource principal available to you and secures the resource principal for you.

Manage Resource Principal using Data Studio

You can also enable Resource Principal using Data Studio.

1. From the Database Actions Launchpad, select the **Data Load** tile.
2. Click **Connections** This opens the Connections page.

3. On the **Connections** page, click **Create** and select **Manage Resource Principal.**



This opens the Manage Resource Principal dialog.

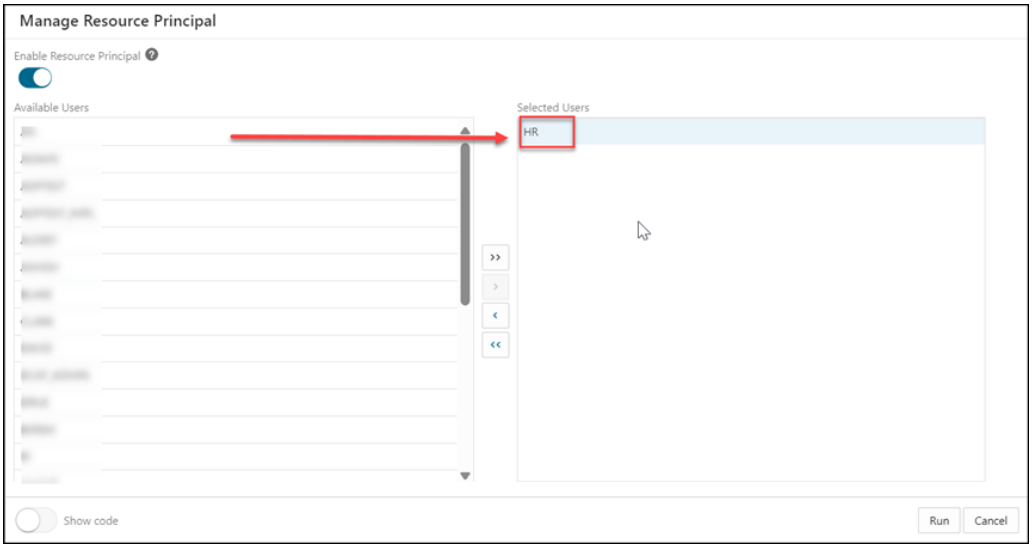
4. Select the user or users you want to add from the **Available Users** list to the **Selected Users** list to enable their resource principal.
Choose any of the available options:

- **>**: This option enables you to move the user to **Selected Users**.

  <: To remove the selected user from **Selected Users**, select this option.

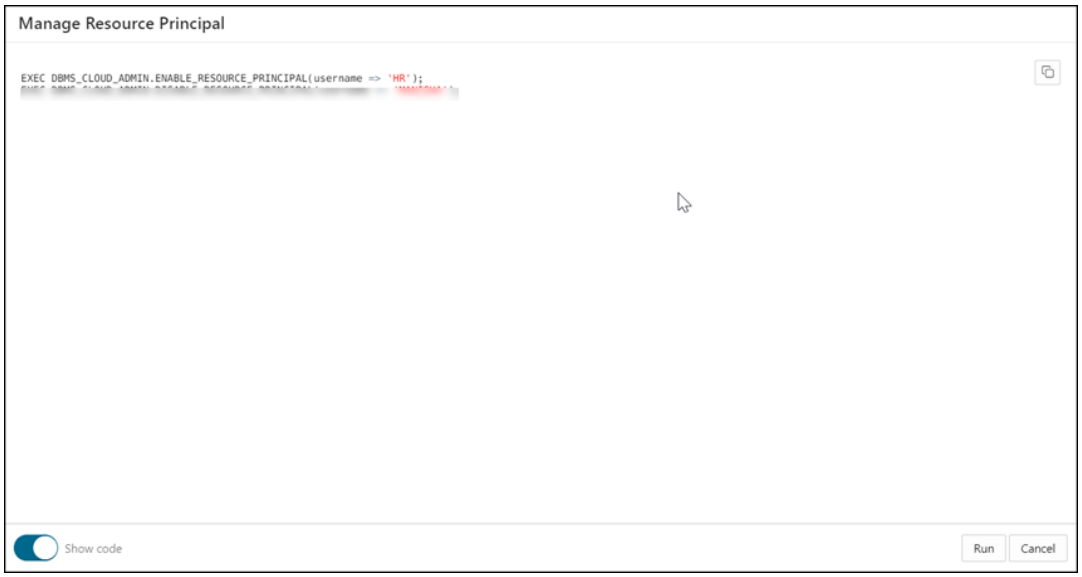
  >>: This option allows you to move all the tables to the **Selected Users** list.

  <<: To remove all the selected users from **Selected Users**, select this option.



> **Note:**
>
> You can grant access to the resource principal credential to a database user only if the ADMIN user has enabled the resource principal credential.

5. Select the **Show Code** option to view the PL/SQL code equivalent of the Manage Resource Principal dialog box. You can copy and execute this PL/SQL code in the worksheet to perform the same action that occurs when you click **Enable Resource Principal** in the Manage Resource Principal dialog box.



6. Click **Run** to complete the process of enabling the resource principal of the selected users. A confirmation notification is displayed that confirms that the resource principal of selected users is enabled.

7. Select a user from Selected Users to Available Users to **disable** the resource principal for the selected user. This removes the credential `OCI$RESOURCE_PRINCIPAL`. Refer to Disable Resource Principal on Autonomous Database for more information on this topic.

Loading Data

You can load data from files on your local device, from remote databases, or from cloud storage buckets from directories and share providers.

The following topics describe the interfaces for these actions.

- Loading Data From Local Files
- Loading Data from Other Databases
- Loading Data from Cloud Storage
- Loading from File System
- Loading Data from AI Source
- Loading from Share Providers
- Use OCI Language Service Capabilities in Data Studio
  You can utilize OCI Language Service Capabilities such as Sentiment Analysis and Key Phrase Extraction to analyze data without machine learning (ML) or artificial intelligence (AI) expertise.
- Use GeoJSON in Data Load
  A GeoJSON object accommodates information about the specific geometry (e.g. Point, Line, Polygon, etc.) along with optional metadata (e.g. ID, etc.).
- Loading Data From Local Files
  To load data from local files into your Oracle Autonomous Database, on the Data Load page, select **LOAD DATA** and **LOCAL FILE**.
- Loading Data from Other Databases
  To load data from tables in another database into your Oracle Autonomous Database, on the Data Load page, select **LOAD DATA** and **DATABASE**. Select a database link from the drop-down list. Drag one or more tables from the list of database tables and drop them in the Data Load Cart.
- Loading Data from Cloud Storage
  You can load data from a cloud store to a table in your Autonomous Database.
- Loading Data from AI Source
  You can use Data Studio tools to load data from AI source.
- Loading Data from File System
  You can load files from file system directories to your Autonomous Database.
- Loading Data from Share
  You can select tables from a Share. You need to subscribe and access provided data share.
- Create Live Feed from Data Load
  The Data load tool loads the data from folders in cloud object stores and enables it to schedule repeated data loads in real time. This is the creation of Live Feed from a data load job.

Use OCI Language Service Capabilities in Data Studio

You can utilize OCI Language Service Capabilities such as Sentiment Analysis and Key Phrase Extraction to analyze data without machine learning (ML) or artificial intelligence (AI) expertise.

For example, you can use it for feedback on a product. A phone manufacturer has launched a new phone model and they want to know what the customer sentiment is on their product. If a large percentage of sentiment is negative it could signal a potential fault with the camera which wasn't detected in Quality Control (QC).

In this section, we will go through the following topics:

- Overview of Sentiment Analysis and Keyphrase Extraction
- Parameters to Analyze Data
- Perform Sentiment Analysis
- Perform Key Phase Extraction

- Overview of Sentiment Analysis and Key Phrase Extraction
  Sentiment Analysis and Key Phrase Extraction is currently supported in loading data from local files and loading data from cloud storage.
- Parameters to Analyze Data
  When you invoke an **Add Expression** from the Settings tab, you must configure the model using parameters.
- Perform Sentiment Analysis
  To determine the Sentiments of input data:
- Perform Key Phrase Extraction
  To extract Key Phrase information from input data:

Overview of Sentiment Analysis and Key Phrase Extraction

Sentiment Analysis and Key Phrase Extraction is currently supported in loading data from local files and loading data from cloud storage.

Sentiment Analysis

Sentiment Analysis analyses the text to define your sentiment on a topic or product. The Language service sentiment analysis uses natural language processing (NLP). The Data Studio tool uses the Oracle Cloud Infrastructure (OCI) Language service to analyze and understand the input data. The Data Studio tool dynamically adds new columns to their data load that contains the output of the OCI Language service. You can detect the sentiments of any column of the source data. For example, when searching through a column containing reviews for an application, assume that you want a general opinion about the application. The Data Studio tool performs sentiment analysis on the input data and creates a new expression column defined in the target table that consists of the sentiment.

For more details, refer to Sentiment Analysis in OCI.

Key Phrase Extraction

Key phrase extraction identifies the main concepts in a text. Keyword extraction is the automated process of extracting the words with the most relevance, and expressions from the input text. It helps summarize the content and recognizes the main topics. A word cloud can be generated with key phrases to help visualize key concepts in text comments or feedback. For example, a Movie review could generate a word cloud based on key phrases identified in their

comments and might see that people are commenting most frequently about the direction, acting, and cinematography staff.

For more details, refer to Key Phrase Extraction in OCI.

Before you start:

- **Load Data from Local Files or Cloud storage:** Load the data you wish to analyze into your Oracle Autonomous Database from Local files or Cloud storage. Make sure the data is loaded to the data load cart without any errors.
- The tools perform sentiment analysis or Key Phrase extraction while you specify settings for the data load job.

Parameters to Analyze Data

When you invoke an **Add Expression** from the Settings tab, you must configure the model using parameters.

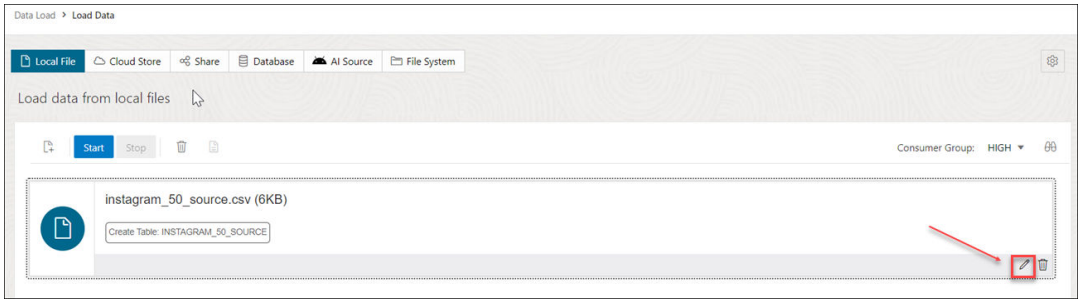
Table 3-3 Parameters for Sentiment Analysis and Key Expression

Parameter	Description
Expression Type	Sentiment Analysis or Key Phrase extraction
Input Column	Select the column which you wish to analyze. The input column drop-down only contains columns that Sentiment Analysis or Key Phrase Extraction supports. For Sentiment Analysis, only VARCHAR2, NVARCHAR2, CLOB, or NCLOB target columns will be displayed in the input drop-down.
Target Column	• Enter the name of the newly created expression column defined in the target table. • For Sentiment Analysis, this displays the sentiment of the input column. The different types of sentiments the tool identifies are: – Positive – Neutral – Mixed – Negative If the tool cannot determine the sentiment of the input column, it returns `NULL` in the expression column. • For Key Phrase Extraction, this displays the key phrases of the input column you select.

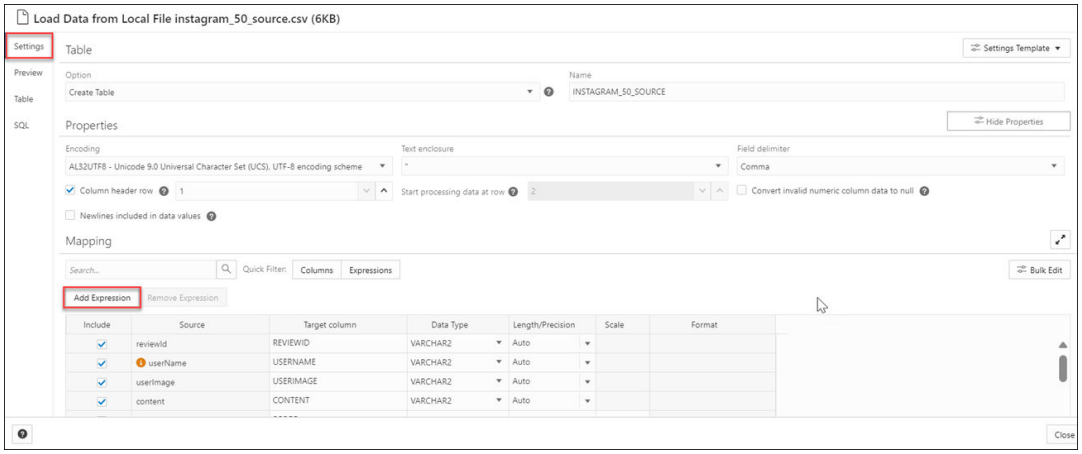
Perform Sentiment Analysis

To determine the Sentiments of input data:

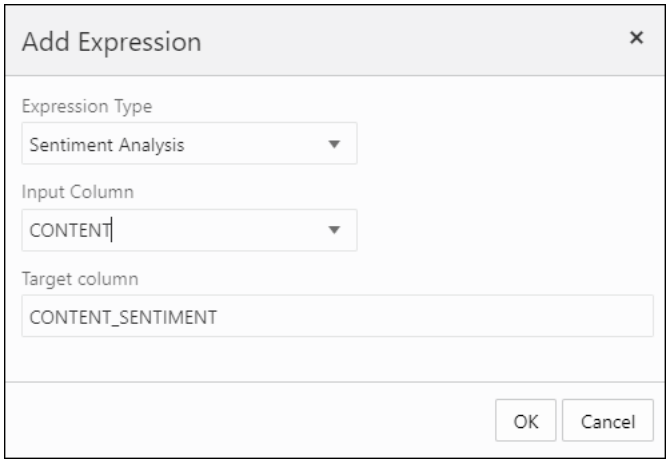
1. After you load data into the Data Load cart from local files or cloud storage, you can view the file in the cart. Click the **settings** icon.



2. Clicking the Setting icon opens a **Load Data from Local File** wizard. In this example, we have loaded data from a local file.
3. On the Settings tab of the wizard, click **Add Expression** under the Mapping section.



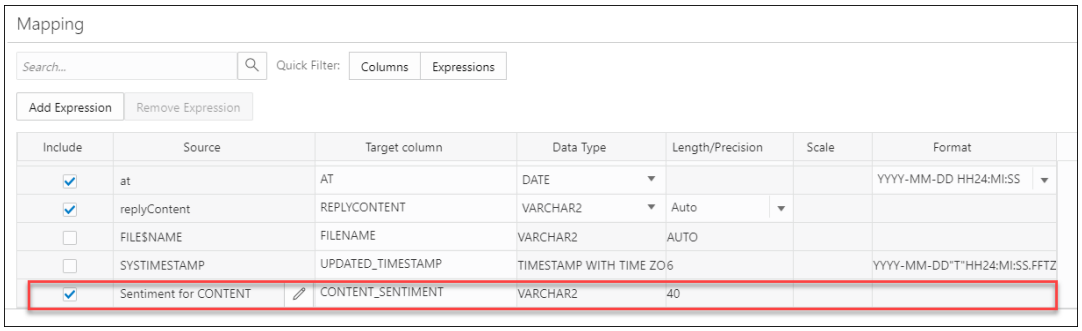
This opens the **Add Expression** dialog box.



4. On the **Add Expression** dialog, specify the following fields:
   - **Expression Type:** From the Expression Type drop-down, select **Sentiment Analysis**.
   - **Input Column:** Select the column from the drop-down that you wish to analyze. For example, *CONTENT*.

- **Target column:** Enter the name of the newly created expression column. For example, *CONTENT_SENTIMENT.*
  Refer to the Parameters to Analyze Data for more details.

5. Click **OK**. You will see a new row added to the mapping grid. This row determines the output expression column generated by the OCI Language service.



6. Click **Close**.
7. Click **Start** in the Data Load menu cart to load data from local files. You will view a confirmation message asking if you wish to start the load from local files.
8. Click **Run** to confirm.

When the data load job completes, the **Table and View Loads** of the Data Load page display the details of the source table that is loaded into the tool. It displays the number of rows and columns and when the upload was complete.

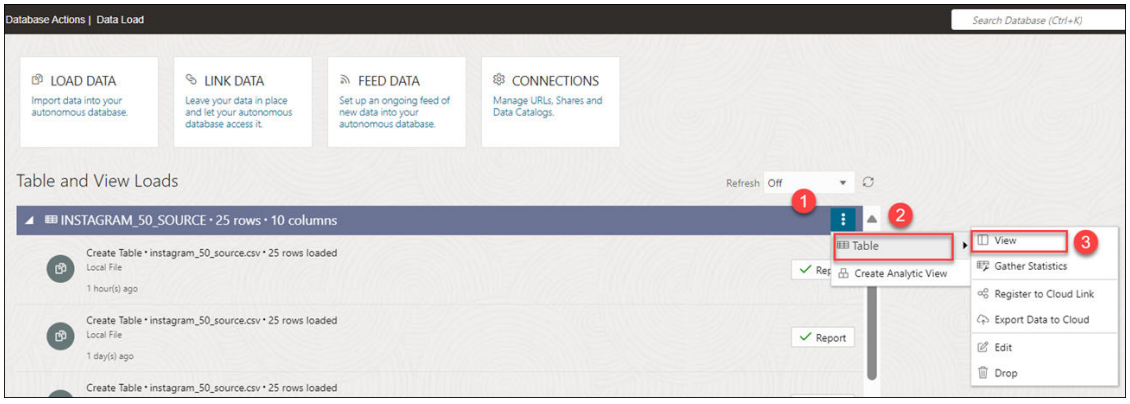
- Output Data Generated from OCI Sentiment Analysis
  When you analyze columns using OCI Language Service model, the Data Studio generates a new expression column and saves the result in the updated table.

Output Data Generated from OCI Sentiment Analysis

When you analyze columns using OCI Language Service model, the Data Studio generates a new expression column and saves the result in the updated table.

To locate the generated expression column, from the Database Actions Launchpad, navigate to **Data Load**. Select the table you load under the **Table and View Loads** section.

Click the three vertical dots beside the load name, and click **Table** then select **View**.



This opens the Preview tab of the data load which displays the updated source file. For example, here is an output dataset from sentiment analysis of the Instagram application. Here, `CONTENT` is the target column, and `CONTENT_SENTIMENT` is the sentiment analysis of the

input column. This column displays one of the following values such as positive, neutral, mixed, or negative It displays **Null** when the tool is unable to determine the sentiment.

INSTAGRAM_50_SOURCE

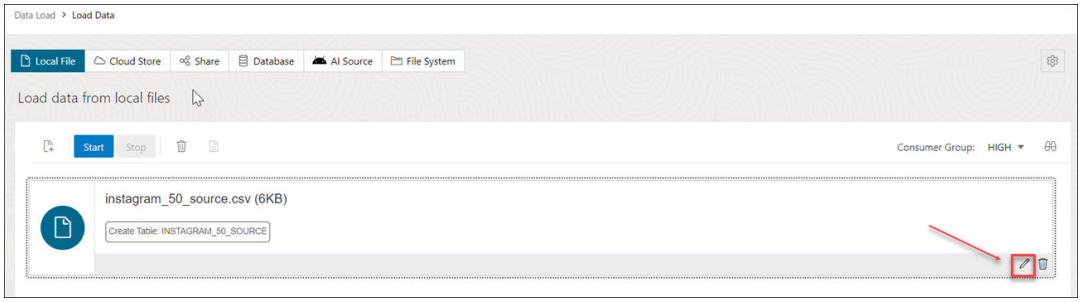
Preview
Lineage
Impact
Statistics
Job Report
Data Definition

		CONTENT	CONTENT_SENTIMENT
1	[redacted]	Nice app pppp	Positive
2	[redacted]	This is very good app	Positive
3	[redacted]	Osm	(null)
4	[redacted]	Mast app aaye 👍	(null)
5	[redacted]	Good	Positive
6	[redacted]	Osm	(null)
7	[redacted]	Good as a social media	Positive
8	[redacted]	Good	Positive
9	[redacted]	Best	Positive
10	[redacted]	💯 💯 💯 👍 👌 👌 👌	(null)
11	[redacted]	Amazing	Positive
12	[redacted]	I really enjoy using this app	Positive

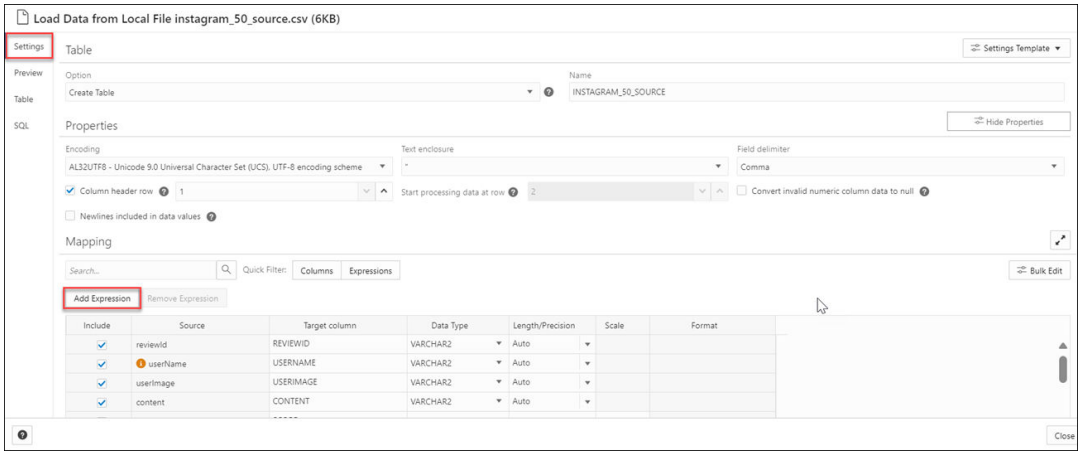
Perform Key Phrase Extraction

To extract Key Phrase information from input data:

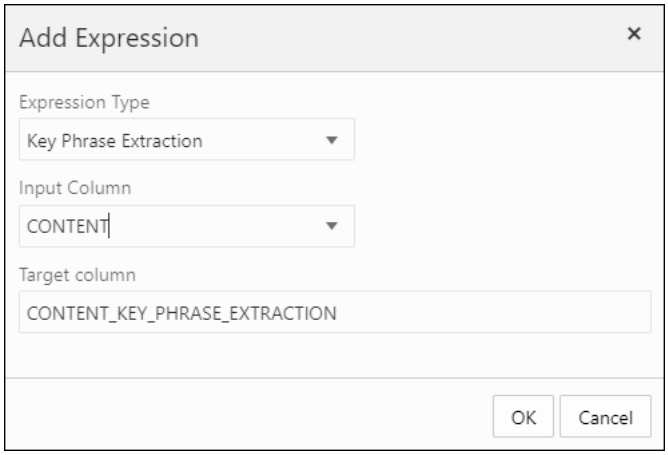
1. After you load data into the Data Load cart from local files or cloud storage, you can view the file in the cart. Click the **settings** icon.



2. Clicking the Setting icon opens a **Load Data from Local File** wizard. In this example, we have loaded data from a local file.
3. On the Settings tab of the wizard, click **Add Expression** under the Mapping section.

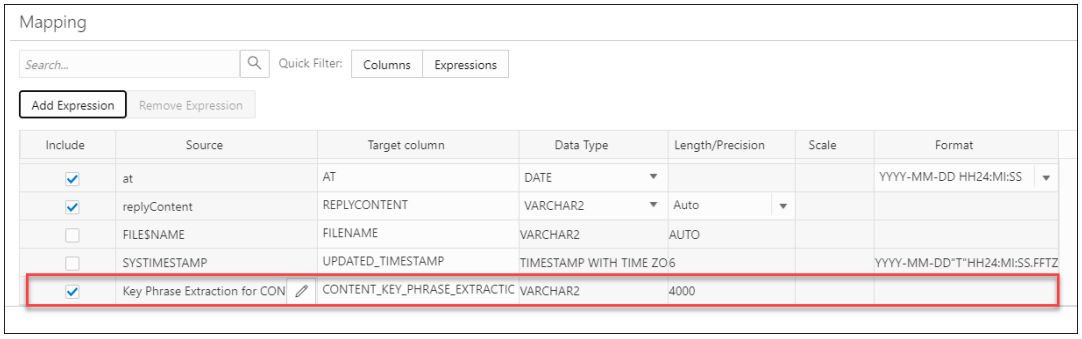


This opens the Add Expression dialog box.



4. On the Add Expression dialog, specify the following fields:
   - **Expression Type:** From the Expression Type drop-down, select Key Phrase Extraction.
   - **Input Column:** Select the column from the drop-down that you wish to analyze. For example, *CONTENT*.
   - **Target column:** Enter the name of the newly created expression column. For example, *CONTENT_KEY_PHRASE_EXTRACTION.*

   Refer to the Parameters to Analyze Data for more details.
5. Click **OK**. You will see a new row added to the mapping grid .This row determines the output expression column generated by the OCI Language service.



6. Click **Close**.
7. Click **Start** in the Data Load menu cart to load data from local files. You will view a confirmation message asking if you wish to start the load from local files.
8. Click **Run** to confirm.

When the data load job completes, the **Table and View Loads** of the Data Load page display the details of the source table that is loaded into the tool. It displays the number of rows and columns and when the upload was complete.

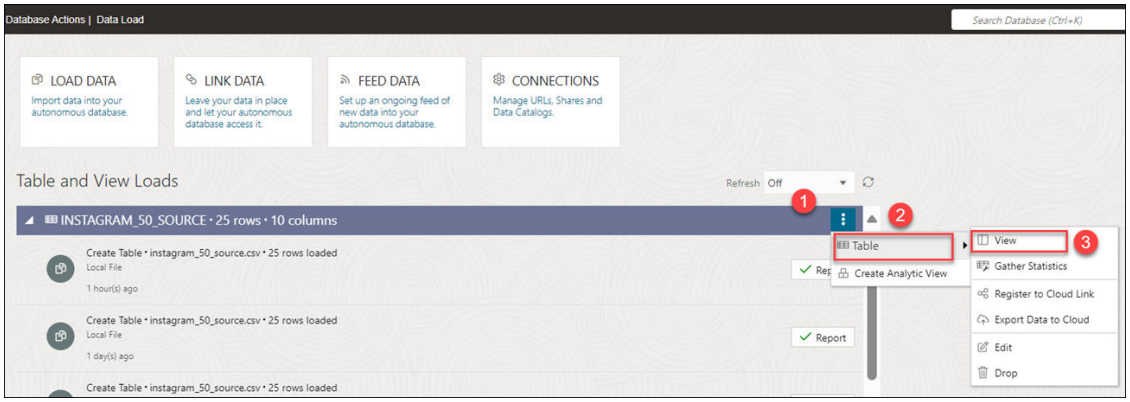
- Output Data Generated from OCI Key Phrase Extraction
  When you analyze columns using OCI Language Service model, the Data Studio generates a new expression column and saves the result in the updated table.

Output Data Generated from OCI Key Phrase Extraction

When you analyze columns using OCI Language Service model, the Data Studio generates a new expression column and saves the result in the updated table.

To locate the generated expression column, from the Database Actions Launchpad, navigate to **Data Load**. Select the table you load under the **Table and View Loads** section.

Click the three vertical dots beside the load name, and click **Table** then select **View**.



For example, here's an output dataset from sentiment analysis of the Instagram application. Here, `CONTENT` is the target column, and `CONTENT_KEY_PHRASE_EXTRACTION` column displays the key phrases extracted from the input column.

INSTAGRAM_50_SOURCE

Preview
Lineage
Impact
Statistics
Job Report
Data Definition

	CONTENT	TENT_KEY_PHRASE_EXTRACTION
1	Nice app pppp	"nice app pppp"
2	This is very good app	"good app"
3	Osm	"osm"
4	Mast app aaye 👍	"mast"
5	Good	(null)
6	Osm	"osm"
7	Good as a social media	"social media"
8	Good	(null)
9	Best	(null)
10	💯 💯 💯 👍 👌 👌 👌	(null)
11	Amazing	(null)
12	I really enjoy using this app	"app"
13	Good	(null)
14	Always my first choice. 👍 👍	"first choice. 👍 "
15	Good	(null)
16	I love talking to my friends	"friends"
17	Best social media app in the	"best social media app"

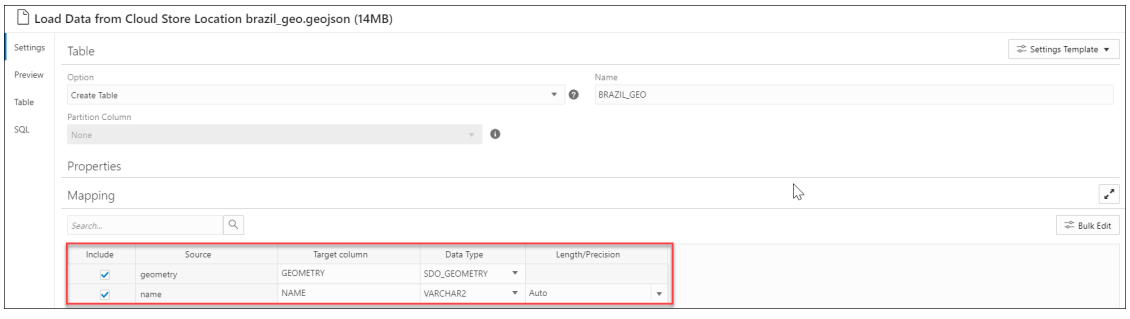
Use GeoJSON in Data Load

A GeoJSON object accommodates information about the specific geometry (e.g. Point, Line, Polygon, etc.) along with optional metadata (e.g. ID, etc.).

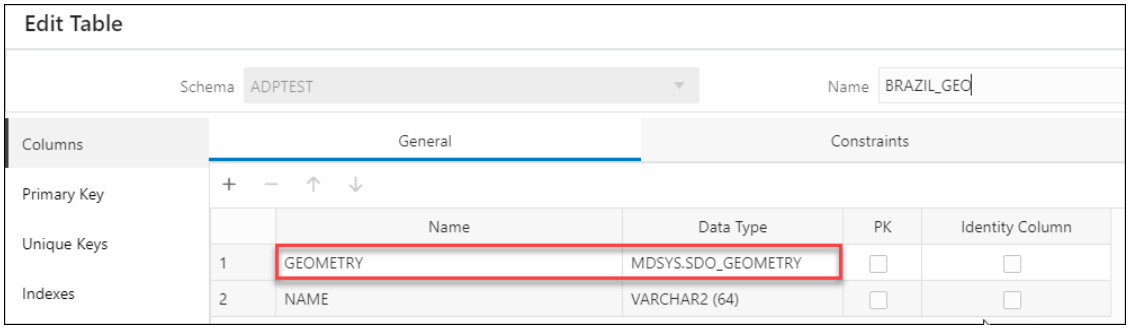
The extension for a GeoJSON file is *.`geojson`. You can load GeoJSON data into the Autonomous Database using the Data Load in Data Studio. If the table contains GeoJSON data, then the data is loaded in a column that projects GeoJSON data from the document set of SQL data type `SDO_GEOMETRY`.

Load a table with GeoJSON Data

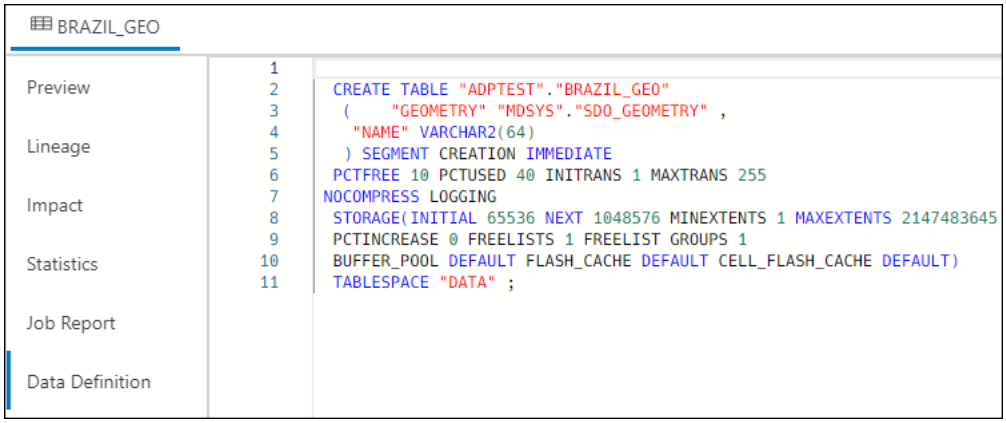
When you load a table in Data Studio with GeoJSON data and review its settings, you will see that it creates table `Brazil_Geo`, which has a column `geometry` of GeoJSON data.



After you load `BRAZIL_GEO` you will view that the tool loads GeoJSON object into a new column `geometry` with the data type `SDO_GEOMETRY`.



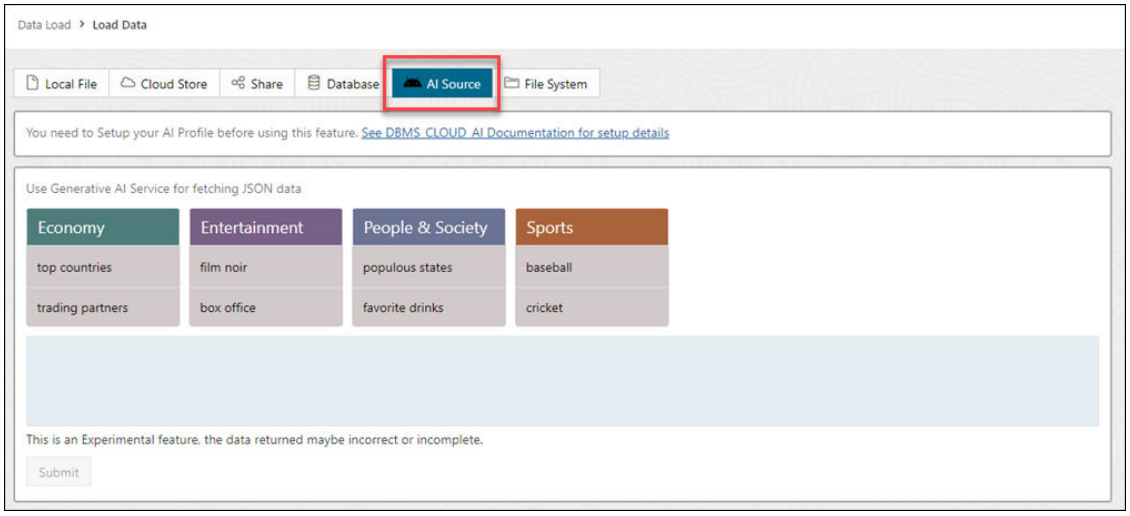
You can also view the same in the **Data Definition** tab when you **View Details** of the Table load after it is run.



Loading Data from AI Source

You can use Data Studio tools to load data from AI source.

On the Data Load dashboard, click **Load Data** card, and select **AI Source**.



You need to perform a pre-requisite of setting up your AI profile before using this feature. See Configure DBMS_CLOUD_AI PackageConfigure DBMS_CLOUD_AI Package for details on the setup of this feature.

Loading Data from File System

You can load files from file system directories to your Autonomous Database.

You can set filters on the data for a table to load only the specified data. For example, to limit the files to only those that are CSV files, enter *.CSV in the file extension filter.

Configure and run a data load job from the Load Cloud Object page. To open that page:

1. On the Data Studio tab and select **Data Load**. You will view the Data Load dashboard.
2. Click **LOAD DATA** and select the **File System** option.

On the top of the page you need to select the directory from where you need to load the files. On the left side of the page is a navigator pane, where you choose the files in the directory containing the data. On the right of the page is the data load cart, where you stage the files and folders for the data load job. You can set options for the data load job before running it. The Autonomous Database comes with predefined CPU/IO shares assigned to different consumer groups. You can set the consumer group to either low, medium or high while executing a data load job depending on your workload. To load files from a directory into your database, do the following:

1. Prepare the Data Load Job: Refer to the Prepare the Data Load Job section for more details.
2. Add Files or Folders for the Data Load Job: Refer to the Add Files or Folders for the Data Load Job section for more details.
3. Enter Details for the Data Load Job: Refer to the Enter Details for the Data Load Job for more details.
4. Run the Data Load Job: Refer to the Run the Data Load Job section for more details.
5. View Details About the Data Load Job After It Is Run: Refer to the View Details About the Data Load Job After It Is Run section for more details.
6. View the Table Resulting from the Data Load Job: Refer to the View the Table Resulting from the Data Load Job section for more details.

- Create Directories in Database Actions
  In the Autonomous Database, there is a preconfigured `data_pump_dir` where you can store files. You can also create directories, drop directories, and attach network file systems.

Create Directories in Database Actions

In the Autonomous Database, there is a preconfigured `data_pump_dir` where you can store files. You can also create directories, drop directories, and attach network file systems.

For example, you can use the `CREATE DIRECTORY` command to create additional directories. Use the database `DROP DIRECTORY` command to drop directories and use `DBMS_CLOUD.LIST_FILES` to list the contents of a directory.

Create a Directory

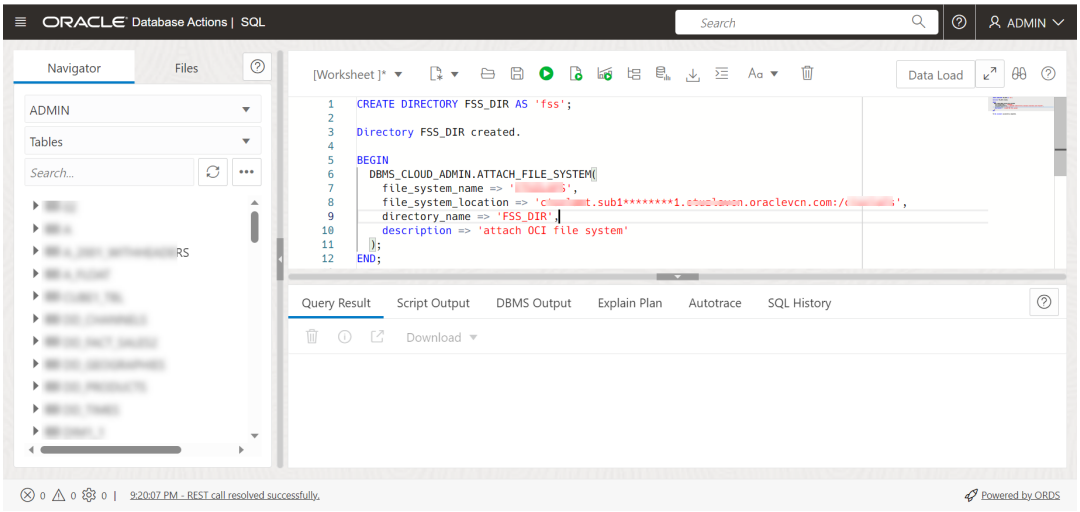
To add a directory, you must have the `CREATE ANY DIRECTORY` system privilege. The ADMIN user is granted the `CREATE ANY DIRECTORY` system privilege. The ADMIN user can grant `CREATE ANY DIRECTORY` system privileges to other users.

See CREATE DIRECTORY for more information.

> **Note:**
>
> - `CREATE DIRECTORY` creates the database directory object in the database and also creates the file system directory. For example the directory path could be:
>
>   `/u03/dbfs/7C149E35BB1000A45FD/data/stage`
>
> - You can create a directory in the root file system to see all the files with the following commands:
>
>   ```
>   CREATE OR REPLACE DIRECTORY ROOT_DIR AS '';
>   ```
>
>   After you create the `ROOT_DIR` directory, use the following command to list all files:
>
>   ```
>   SELECT * FROM DBMS_CLOUD.list_files('ROOT_DIR');
>   ```
>
> - To run `DBMS_CLOUD.LIST_FILES` with a user other than ADMIN you need to grant read privileges on the directory to that user.

Let us demonstrate how to create a directory and access it from Data Studio:

- **Create a directory in Database Actions:**
  Login to your Database Actions instance and select the SQL card under **Development**. You can view the SQL worksheet. Now create a directory and attach a file system name of your choice to the directory you create. In the below given example, `FSS_DIR` is the name of the directory.



  Run the above command. The above command gives the following output:

  `PL/SQL procedure successfully completed.`

- **Attach the file system**

Attach your file system with the name of your choice to the `FSS_DIR` directory via the DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM procedure.

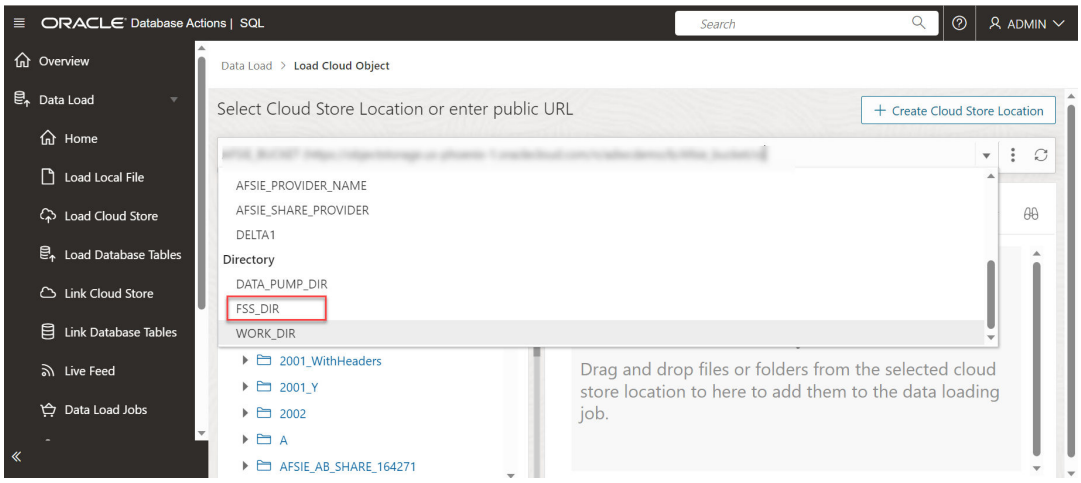
```
BEGIN
  DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM(
    file_system_name => '********',
    file_system_location => '*******.sub1********1.********.oraclevcn.com:/
********',
    directory_name => 'FSS_DIR',
    description => 'attach OCI file system'
  );
END;
/
```

Run the above command to view the following output:

```
PL/SQL procedure successfully completed.
```

- To view the attached file system, run the following command:
  `SELECT file_system_name, file_system_location, directory_path FROM dba_cloud_file_systems;`

  You will view the file system name, file system location and directory path.
- You can view the new directory along with the files attached to it by navigating to Load Cloud Object under Data Load menu of the Data Studio tool. Click the **Select Cloud Store location** drop-down.



You can view the newly created directory `FSS_DIR`. You can load data from the file system directories to the autonomous database using the Data Load tool. See Loading Data from File System.

Loading Data from Share

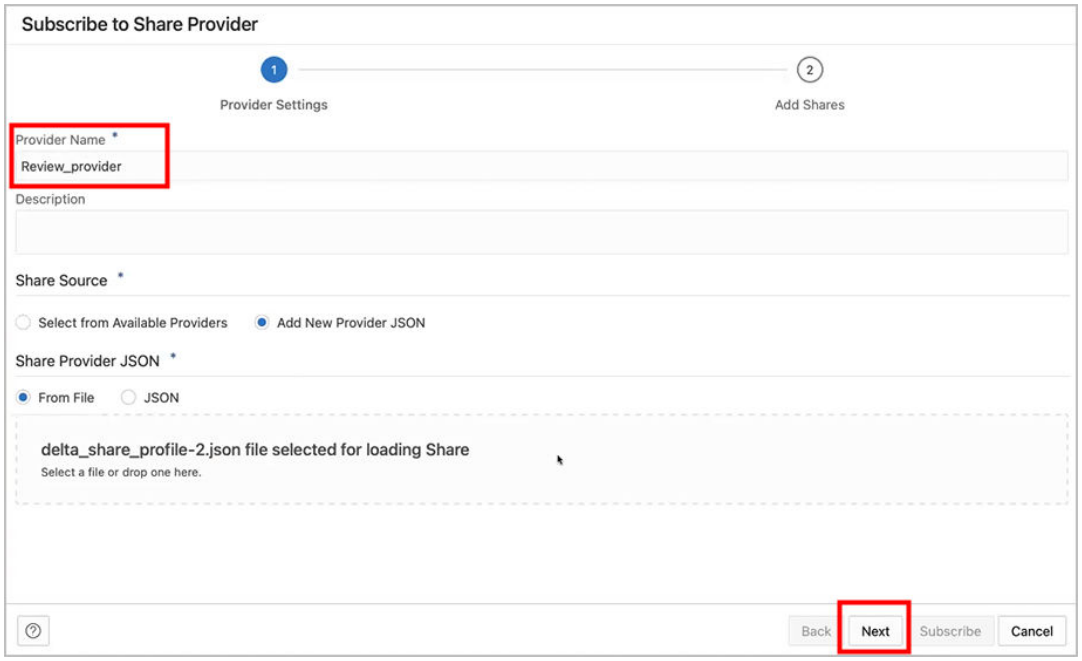
You can select tables from a Share. You need to subscribe and access provided data share.

To load tables from a share, click **Load Data** on the Data Load Dashboard. Click **Share** on the Load Data page. Click **+ Subscribe to Share Provider** to subscribe to a Share Provider.

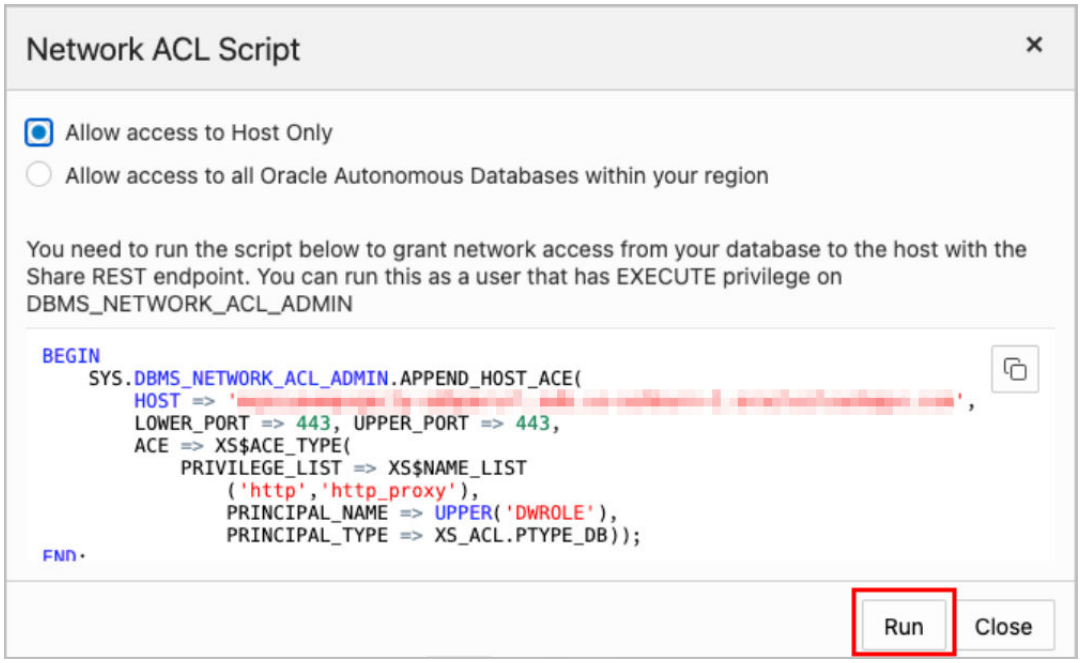
Subscribe to the Data Share Provider

To subscribe, you need to use the information contained in the uploaded JSON profile you received from the share provider.

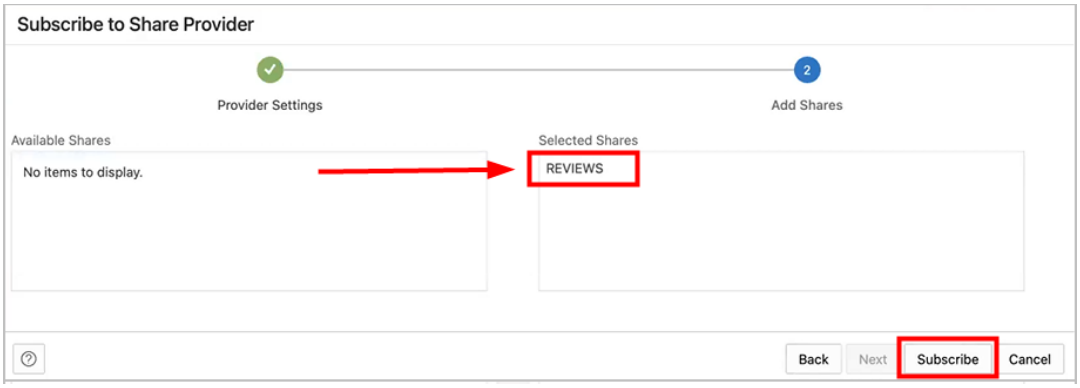
1. On the Subscribe to Data Share Provider, enter the **Provider Name**.



2. Select Add New Provider JSON and click on the Delta Share Profile JSON to drag and drop the JSON profile.
3. Click **Next** to progress to the Add Shares tab.
4. Select the level of network access you want to allow from your database to the host with the Share REST endpoint, and click **Run**. In this example, **Allow access to Host Only** is selected.

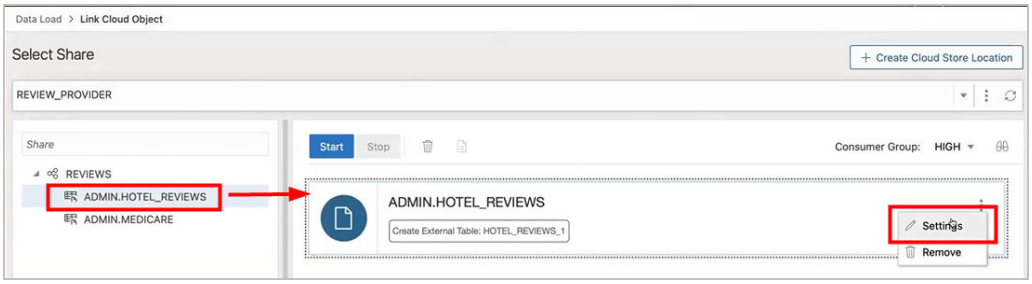


5. To register the shares available to you, move shares from **Available Shares** to **Selected Shares** and click **Subscribe**.
   The screenshot below shows the **REVIEWS** share moved from **Available Share**s to **Selected Shares** before clicking **Subscribe**.

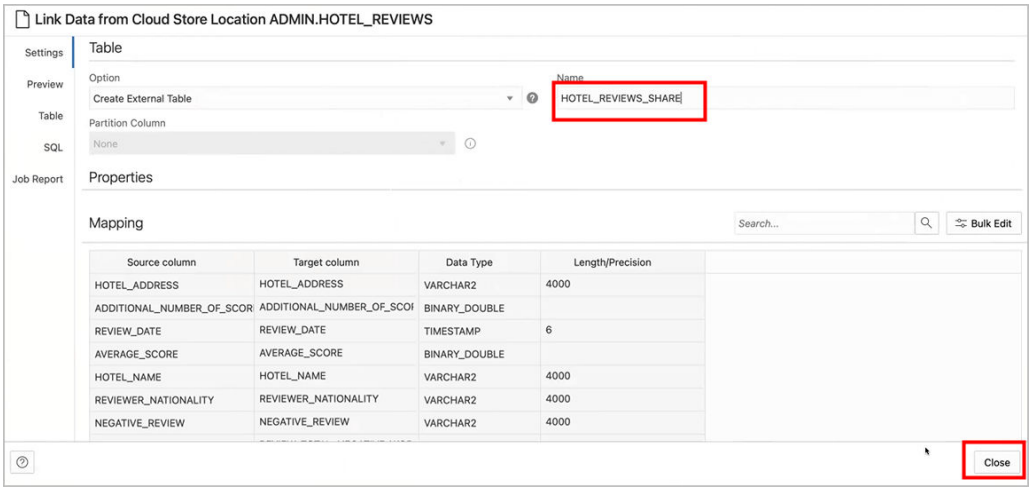


6. Create external tables derived from tables selected from the data share.

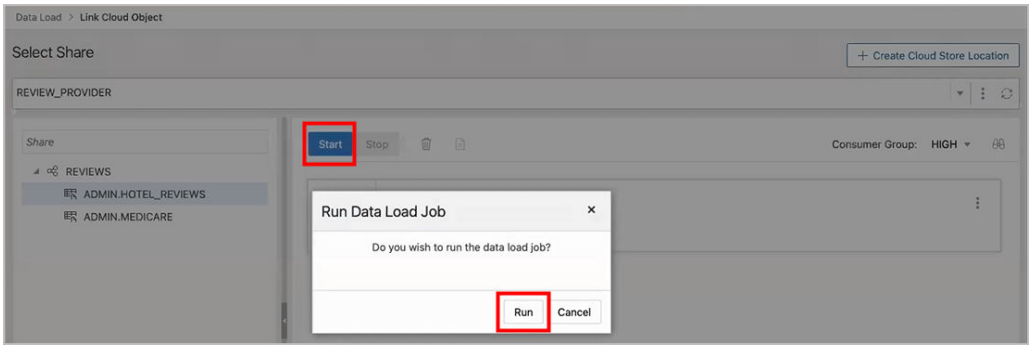
   a. Drag and drop tables from the selected share. You can optionally click **settings** to view the table details.
      In this example, the only table selected is **HOTEL_REVIEWS**.



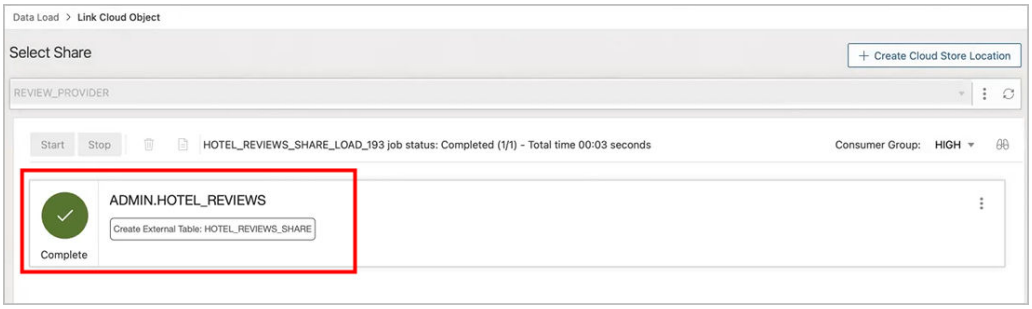
   b. You can optionally change the name of your table and click **Close**.
      In this example, the name is changed from **HOTEL_REVIEWS** to **HOTEL_REVIEWS_SHARE**.



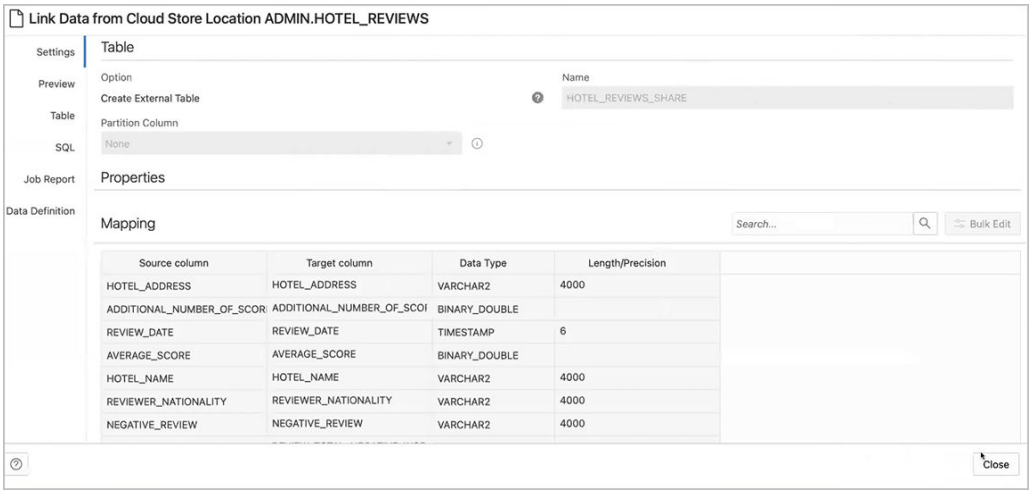
c. Create the external table by clicking **Start**, on the Select Share page, and then clicking **Run** on the Run Data Load Job dialog.



d. When the external tables are created the Complete message is displayed.



e. Click on the external table name to view the table details.

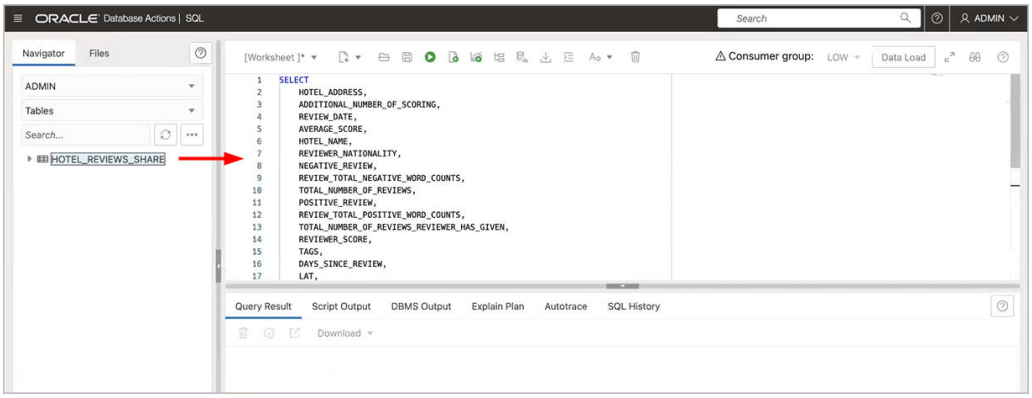


7. View the created tables from Database Actions.

   a. Click on **Database Actions**, in the breadcrumb, to go back to the Database Actions launchpad.

   b. Click on the **SQL** tile.

   c. Select the external table, drag and drop it into the worksheet.
   The SQL Select statement for the table appears. This SQL statement can be run to consume the shared data.



8. Drag and drop tables from selected share

Create Live Feed from Data Load

The Data load tool loads the data from folders in cloud object stores and enables it to schedule repeated data loads in real time. This is the creation of Live Feed from a data load job.

Once the data load is complete, you can create a Live Feed from a cart item that loaded an object store folder using the Create Table or Drop Table and Create New Table options.

To create a live feed from Data Load:

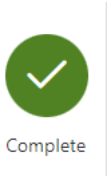
1. Click **Selector** to display the navigation menu. Under Data Studio, select **Data Load**.
2. Select the **Load Data** tile to load data from various sources such as local files, databases, cloud store, directories, and Shares.
3. Click **Cloud Stores**to load objects from URLs or Cloud store links.
4. Select the cloud store location from the drop-down. If are not able to view the cloud store location, select **Create Cloud Store Location** to create a new cloud store location. Follow the steps described in the Create Oracle Cloud Infrastructure Native Credentials if you do not have a cloud location available.
5. After you select the cloud store location, you can view the list of folders and files present in the cloud storage. Add files from the cloud store to the data load cart, where you can edit the details of the load job.

> **Note:**
>
> The Data load tool does not support the creation of a live feed from a loaded cart item that consists of a single file in CSV, XLS, XLSX, TSV, TXT, XML, JSON, and AVRO format or of a folder that contains a file in XLSX format.

6. To add the folders, drag a folder from the file navigator on the left and drop them into the cart on the right. When you add the folder to the cart, a prompt is displayed that asks if you want to load all the objects from the multiple source files into a single target table. Click **Yes** to continue or **No** to cancel. You can add multiple folders to the cart, the data represented by each card will be loaded into a separate table, but all the items in the cart will be processed as part of the same data load job.
7. Select **Settings** (pencil icon) from the data load cart to enter the details about the data load job.
8. In the **Settings** tab of the Load Data from the Cloud Store Location, you can select **Create Table** or **Drop Table and Create New Table** from the **Option** drop-down.

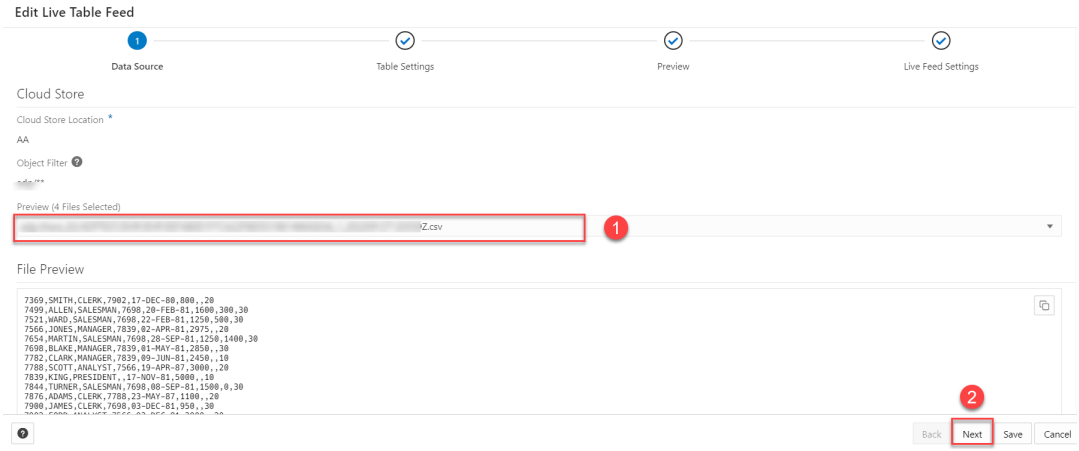
> **Note:**
>
> The Live feed tool works with the Data load job only if you create a table and insert the data into a new table or drop the existing table and insert the data into a new table.

9. Enter the other details for the data load job. For more information on entering the details, refer to the Enter Details for the Data Load Job topic.
10. Once you have added data sources to the data load cart and entered details about the data load job, select **Start** to run the job.
11. After the data load job is run, the data load cart displays a green check mark

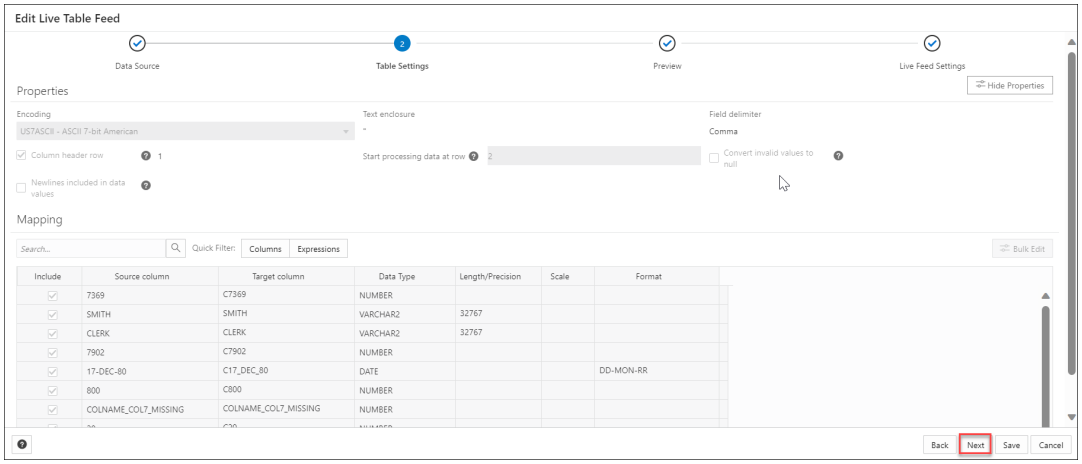


which indicates that the data load job is complete.

12. Click **Create Live Table Feed** on the data load cart to create a Live table feed from the data load job. You will view a successful message that says the creation of Live Table feed is successful and if you wish to edit the live table feed. Click **Yes** to continue and **No** to cancel. Selecting **Yes** opens an Edit Live Table Feed wizard.
13. On the Edit Live Table Feed wizard, you can view the **Cloud Store Location** of the source folder and the **Object Filter** applied to the data.
Select any file whose data you want to preview from the Preview drop-down in the Data Source tab: The field shows the total number of files present in the cloud store folder you loaded. A preview of the data is displayed.



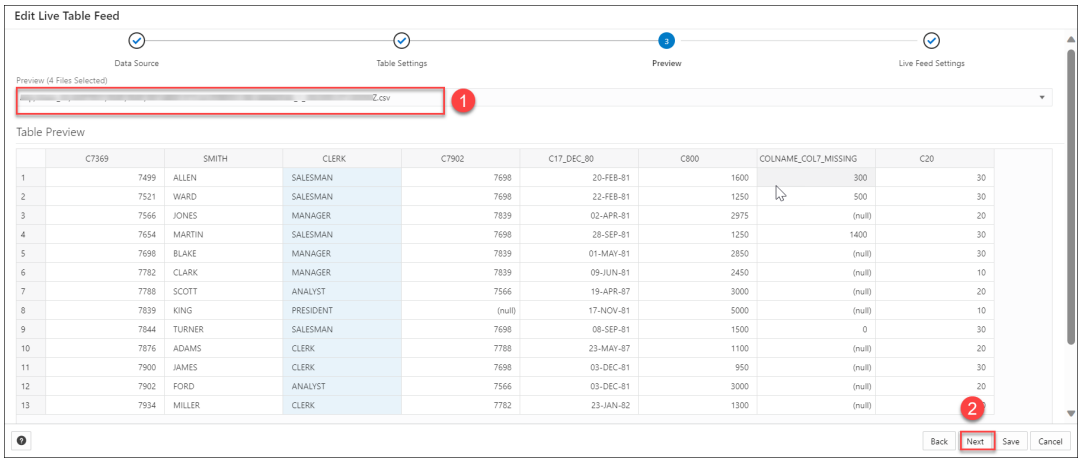
14. Click **Next** to progress to the Table Settings tab.



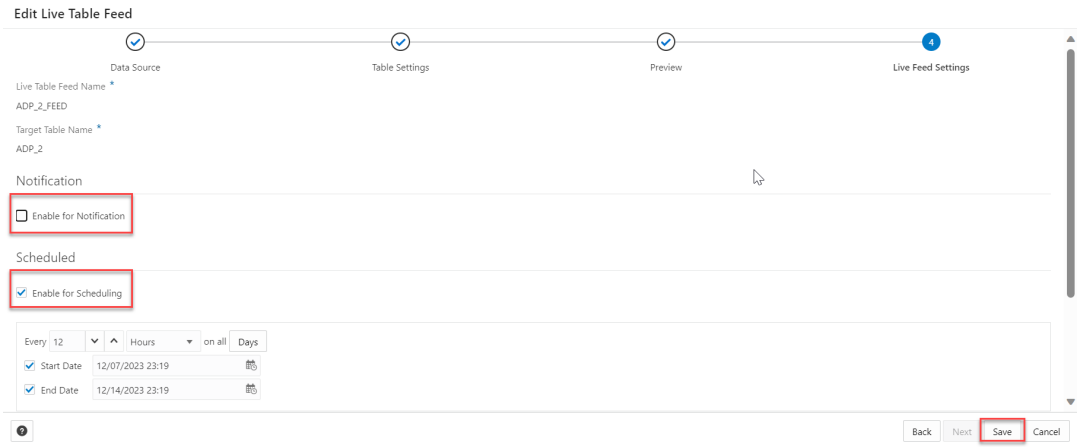
You can view the Properties and Mapping details of the data load job on the Table Settings tab.

> **Note:**
>
> You cannot select or edit any of the details of this tab.

15. Click **Next** to progress to the Preview tab of the wizard.
Select any file from the Preview drop-down to view the file. The Table Preview displays the preview of the file you select from the drop-down.



16. Click **Next** to view the Live Feed Settings tab of the wizard. On the Live Feed Settings tab, specify the following values in the field:



- **Enable for Notification**: Select this option so that new or changed data in the data source will be loaded based on an Oracle Cloud Infrastructure notification. When you select this option, you can avoid delays that might occur when polling is initiated on a schedule (that is, if you selected the live table feed Scheduled option.

  When you select the Enable for Notification option, you must also configure your object store bucket to emit notifications

- **Enable For Scheduling**: Select this option to set up a schedule for data feed. In the time interval fields, enter a number, and select a time type and the days on which to poll the bucket for new or changed files. For example, to poll every two hours on Monday, Wednesday, and Friday, enter **2**, select **Hours**, and then select **Monday**, **Wednesday**, and **Friday** in the appropriate fields. You can select **All Days**, **Monday to Friday**, **Sunday to Thursday**, or **Custom** from the Week Days drop-down. The Custom field enables you to select **Monday**,**Tuesday**, **Wednesday**, **Thursday** and **Friday** .

Select a start and end date. If you don't select a start date, the current time and date will be used as the start date. The end date is optional. However, without an end date, the feed will continue to poll.

The rest of the fields displayed in the wizard such as the **Live Table Feed Name**, **Target Table Name** , and **Consumer Group** are greyed out and disabled for selecting or editing.

Click **Save** to save and create a Live Table Feed from a data load cart.

Loading Data From Local Files

To load data from local files into your Oracle Autonomous Database, on the Data Load page, select **LOAD DATA** and **LOCAL FILE**.

Drag one or more files from your local file system navigator and drop them in the Data Load Cart. You can also click **Select Files** or the Select Files icon, select one or more files from the file system navigator, and then click **Open**.

You can add files in these file formats: AVRO, CSV, JSON, GeoJSON, TSV, delimited TXT, XLS, XLSX, XML. For information on supported file formats, see Format Specifications for JSON, AVRO, and XML Files.

An item for each file appears in the cart. For an XLS or XLSX spreadsheet, the worksheets of the spreadsheet appear as individual items. The item shows the name of the source file or worksheet and its size, and the name of the table that is the target for the data load. The Data Load tool supports loading tables from only the first worksheet of a multi-worksheet XLSX file when the file is in an object store.

You can add more files to the cart by clicking the Select Files icon. You can add any number of files to the cart and load data from all of them in a single data load job.

To remove a source file from the Data Load Cart, click the Remove (trash can) icon for the source item. To remove all source files from the cart, click the Remove All (trash can) icon in the Data Load Cart menu bar.

To return to the Data Load page, click **Data Load** above the Data Load Cart menu bar.

Specify Processing Options

To specify settings for the data load job or preview the data in the source or the target, select the Settings (pencil) icon for the item in the Data Load Cart.

In the settings pane, on the **Settings** tab, you can view the name and size of the file in the title of the Load Data dialog box.

The **Name** field specifies the name of the target table. The value in the field varies depending on the selection in the **Options** field. If the option is **Create Table**, then the default target value is the name of source file or worksheet.

To specify a different name for the target table, enter it in the **Name** field. For the other target table choices in the **Options** field, the default value is <None>. Expand the drop-down list and select an existing table as the target.

In the **Options** field select **Create Table**, **Insert into Table**, **Replace Data**, **Drop Table and Create New Table**, or **Merge into Table**. Point to the question mark icon to see a brief description of the selected option.

Select a different schema from the **Schema** drop-down to create your target table in another schema.

> **Note:**
>
> The **Schema** drop-down is available only if you have `PDB_DBA` role grant to you. To grant yourself a `PDB_DBA` role, you must log into your Database Actions instance and enter the following command in the **SQL** worksheet area displayed in the **SQL** tab under **Development** tools present in the Launchpad.
>
> ```
> Grant PDB_DBA to Username;
> ```
>
> This drop-down is available for **Create Table** and **Drop Table and Create New Table** options.

The **Source column name** option specifies whether to get the source and target column names from the file or to specify the column names manually. Getting the column names from the header of the source file is the default. If you select the **Column header** option, then the first row in the file is processed as column names. If you deselect the option, then the first row is processed as data. To specify column names manually, enter a name for each target column in the **Mapping** section. You can also select a data type for the column.

The **Start processing data at row** field specifies how many rows to skip when loading the source data into the target. If you have selected the **Column header** option, and if you enter a number greater than 0 in the **Start processing data at row** field, then that number of rows after the first row are not loaded into the target. If you have deselected the **Column header**

option, and if you enter a number greater than 0 in the **Start processing data at row** field, that number of rows including the first row are not loaded into the target.

To change the character set encoding for the contents of the file, select a value from the **Encoding** drop-down list.

To specify the characters that enclose text, select the double-quotes or single-quote character or `None` from the **Text enclosure** drop-down list.

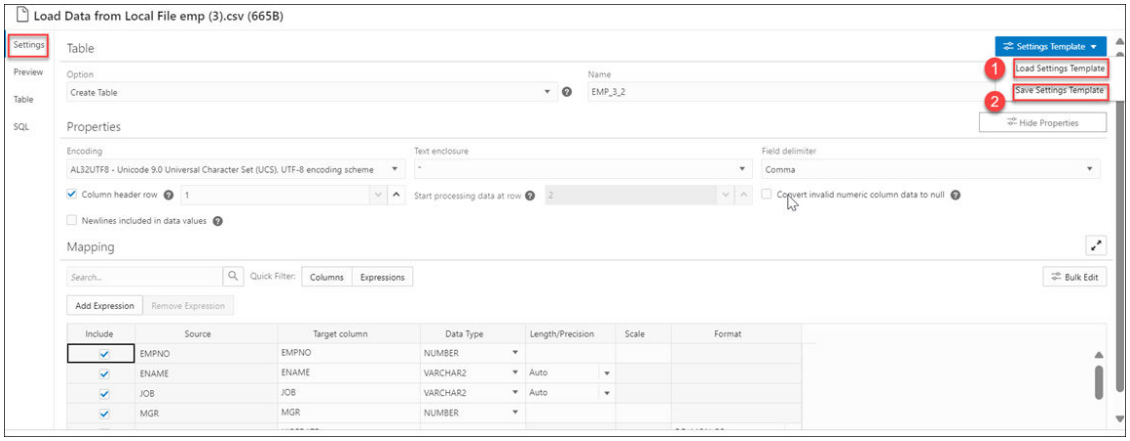
To change the delimiter character that separates columns in the source, expand the **Field delimiter** drop-down list and select a character. For example, if the file has columns delimited by semicolons, change the delimiter from the default comma delimiter to a semicolon.

To convert any invalid value in a numeric source column to a null value in the target column, select the Numeric column **Convert invalid data to null** option.

Settings Template

The save settings feature saves the configuration set in the Cart settings in the form of a JSON file. When opening the Settings template, you have the following options:

1. **Load Settings Template**: Loads a settings template from your local system.
2. **Save Settings Template**: Saves the current existing settings template.



You can use the **Load Settings Template** if you want to use an existing customized template present in your local.

1. From the Settings Template in the Settings tab of the Load Data page, select **Load Settings Template**.
2. You will see a Load Settings Template wizard, click the Settings Template JSON to load a JSON file from your system.
3. Clicking the Settings template JSON will open your local system. Click **OK** to load the JSON file.
4. After you load the JSON file, you can view the updates applied automatically to the settings tab which matches the JSON settings template you load from your local.

You can use the **Save Settings Template** to save the existing current Settings template.

1. From the Settings Template in the Settings tab of the Load Data page, select **Save Settings Template**.
2. The Template file editor appears. Click the Template File name and name the new template.
3. Click **OK** to finish saving the new name of the existing template.

4. You can test the configuration of the new template.

Bulk Edit Settings

You can use the Bulk edit settings to update all the columns at once from the mapping table. Use it to apply changes to the selection currently displayed in the results pane. You can search for the values of the column you want to edit in the search field and click the magnifier icon. The mapping table will display the results of the search. Select the Bulk Edit setting to update the column. The Bulk Edit setting allows you to:

- Update values of all the fields in a group.
- Find and replace, Add Prefix and Add suffix to target column name.
- Include the column(s) for loading data to the target table.
- Exclude the column(s) for loading data to the target table.

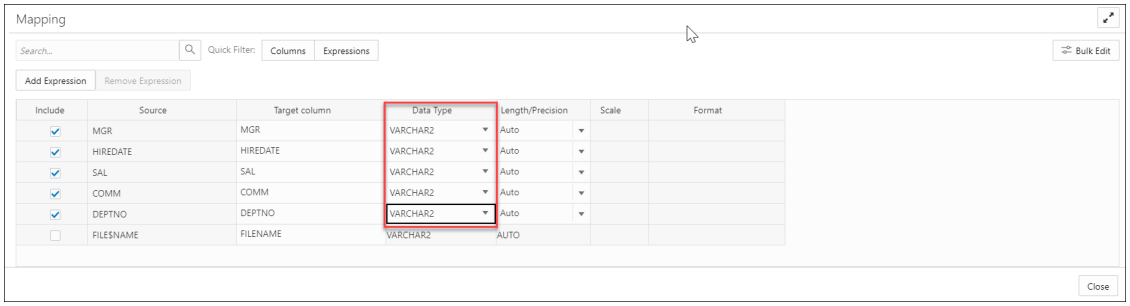
Searching the Column

The Bulk Edit setting updates the columns returned by the search field. The search box besides the Bulk Edit setting icon filters the list of columns you wish to update in a bulk. As soon as you start typing in the search field, the tool returns the field values which contains the letters you type. You can remove the filter by deleting all the content from the search box and clicking the magnifier icon that appears next to the search box.

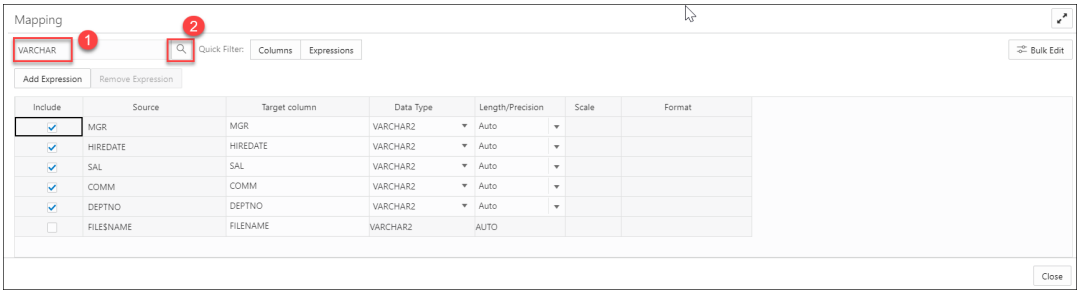
The Bulk Edit setting enables you to update the values of the following columns for all the searches returned by the search field:

- Data Type
- Target Column name
- Include Columns for loading
- Exclude columns for loading

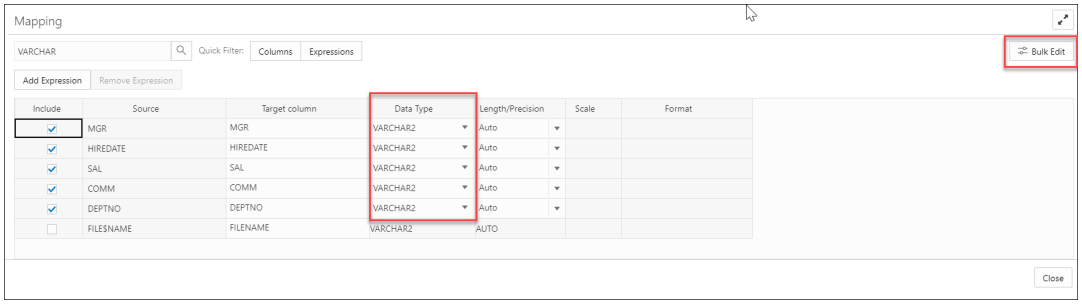
Consider changing the Data Type of first to fifth rows from VARCHAR to NUMBER in the mapping table.



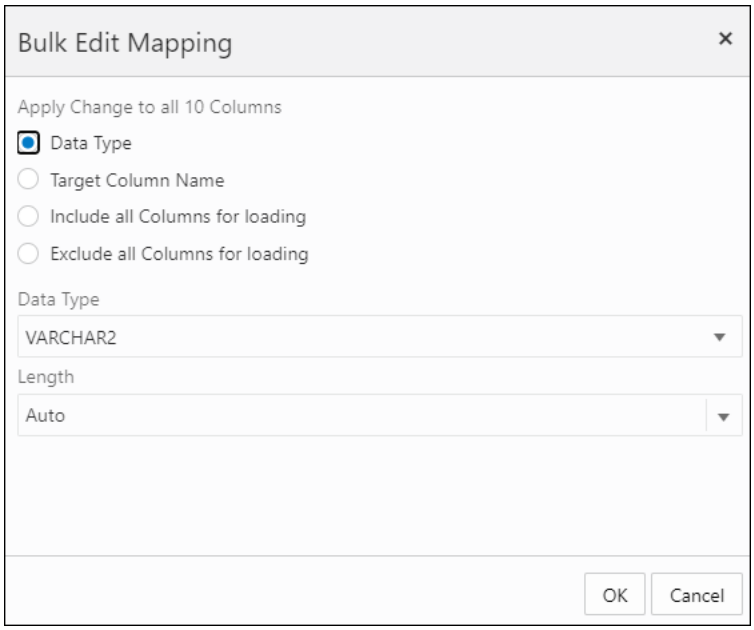
- Type `VARCHAR` in the search field and click on the magnifier icon besides the search field.



- The search will return the rows having `VARCHAR` as the Data Type. Select the **Bulk Edit** icon besides the magnifier icon.

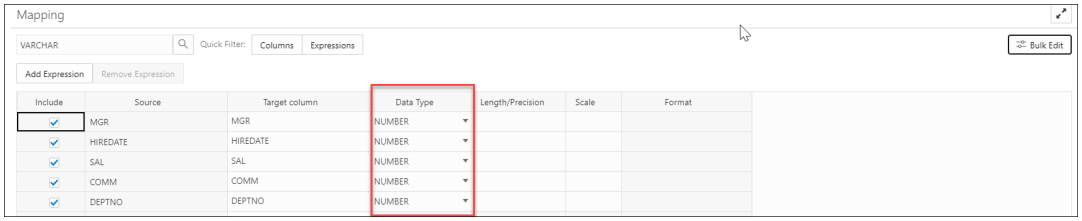


- Selecting the Bulk Edit icon opens the Bulk Edit Mapping dialog box.

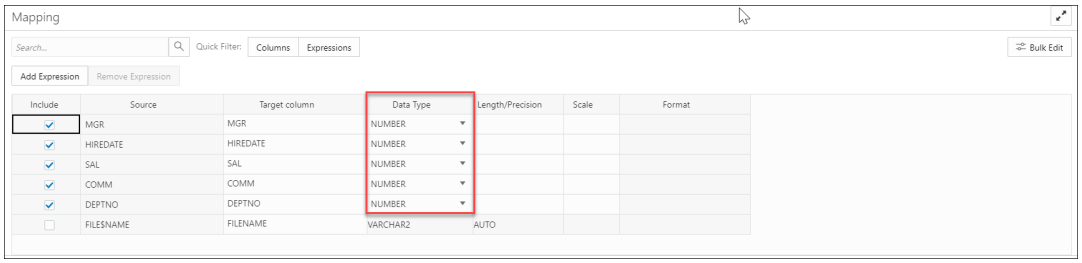


- Click **Data Type** and select `NUMBER` from the Data Type drop-down. Click **OK**.

- The Data Type of the selected rows changes from `VARCHAR` to `NUMBER`.



- Clear the content from the search field and click the magnifier icon to view the bulk edit updates in the Mapping table.



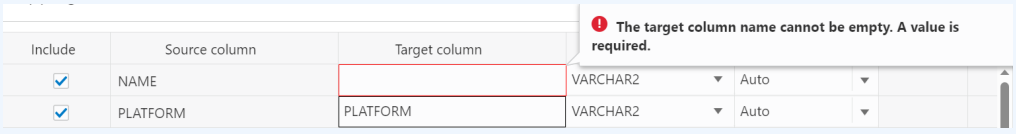
Specify Mappings

You can filter the results in the mapping table with Quick Filter field that enables you to filter out **Columns** or **Expressions**.

Select **Add Expression** to perform **Sentiment Analysis** or **Key Phrase extraction** with the source data. See Use OCI Language Service Capabilities in Data Studio for more details.

If you select the **Create Table** or **Drop Table and Create New Table** option and you are getting the source column names from the file header, then in the **Mapping** section either accept the default values for the target columns and data types or specify different values. To specify different values, in the target column, enter a name for the column. In the Data Type column, select a data type from the drop-down list. If you are not getting the source column names from the file header, then for each source column specify a name for the target column and select a data type from the Data Type drop-down list. For the Date data type, select a date format from the Format drop-down list.

> **Note:**
>
> You will receive a tooltip error message with the exact reason for the error message when you complete editing a cell. The mapping grid cell will be highlighted with red to indicate an invalid value that must be fixed. The highlight is removed after you fix the invalid value. For example, you can view the following tooltip error message when the target column name is not filled in.



For the **Merge into Table** option, for each source column, select a target column form the drop-down list. You must specify at least one column as a key column. To specify a column as a key column, select the **Merge Key** check box for the column. Merge keys are one or more columns that uniquely identify each row in the table. Merge keys must not contain any null values. For loading tables with primary keys, this option automatically enables the selection of primary key columns as the merge keys.

For the **Insert into Table** or **Replace Data** options, for each source column, select a target column from the drop-down list of existing columns.

Preview Source Data

To view a selection of data in the source file, select the **Preview** tab. The Preview tab displays a File menu which displays the data in tabular format with it's values. You can copy the table. The **Load Preview** tab displays the source data.

Any modifications you make in the source preview do not affect the loading of data from the file.

Preview Target Data

For all options except **Create Table**, to view the existing data in the target table, select the **Load Preview** tab. The load preview displays the data in the target table before you run the data load job.

To close the settings pane, click **Close**.

Run the Data Load Job

When you have added all of the sources for the job and specified the settings for each source, to run the job click the Start icon in the Data Load Cart menu bar. In the Run Data Load Job dialog box, click **Start**. To stop the data load job, click the Stop icon.

Once the data load job starts, you can view the progress of the job in the Data Load dashboard. See The Data Load Page for more details.

When the data load job completes, the Data Load dashboard page displays the results of the job. At the top of the header of the table load, you can view the name of the table along with the total columns present in the table.

Click **Report** to view the total number of rows processed successfully and the count of rejected rows. You can also view the Start time. The SQL pane of the Report displays the equivalent SQL code of the job.

To view information about an item in the job, click the **Actions** icon on the Table Load.

To view a log of the load operation, click the Logging icon. You can save the log, clear it, or refresh it. Click **OK** to dismiss the log.

The list of tables on the Data Load page contains any new tables created. The target tables for the **Insert into Table**, **Replace Data**, **Drop Table and Create New Table**, and **Merge into Table** options contain the loaded data.

Fixing a data load job

After your data load job, you might see errors that you want to correct, or upon inspection, realize that you wanted to name a column differently. In such cases, you will view a warning sign on the Table load. Click the **Reload** icon to reload source with suggested fixes. Click **Actions** icon on the Table load and select **Edit** to make any changes to the data load job (i.e., change a column name).

Click **Apply** to apply any changes. Click **Close** to return to the Database Actions page.

Loading Data from Other Databases

To load data from tables in another database into your Oracle Autonomous Database, on the Data Load page, select **LOAD DATA** and **DATABASE**. Select a database link from the drop-

down list. Drag one or more tables from the list of database tables and drop them in the Data Load Cart.

Each table appears as an item in the Data Load Cart. The item shows the name of the table and the number of rows in it, and the name of the table that is the target for the data load.

To remove a table from the Data Load Cart, click the Remove (trash can) icon for the item. To remove all tables from the cart, click the Remove All (trash can) icon in the Data Load Cart menu bar.

To add a remote database to the list of database links, create a database link to the remote database. For information on creating a database link, see Database Links in *Oracle Database Administrator's Guide*.

The databases available to you appear in the drop-down list of the database navigation pane of the Load Tables page.

You can filter the tables displayed in the navigation pane by entering a case-sensitive value in the search field at the top of the navigation tree and pressing Enter. To display all of the tables again, clear the search field and press Enter.

You can add any number of tables from the navigation pane to the Data Load Cart and load data from all of them in a single data loading job. You can set filters on the data for a table to load only the specified data.

Specify Processing Options

To specify settings for the data load job, preview the data in the source or the target, and see statistics about the data, click the Settings (pencil) icon for the item in the Data Load Cart.

In the settings pane, on the Settings tab, you can view the name and size of the file in the title of the Load Data dialog box.

The **Table** field specifies the name of the target table. The value in the field varies depending on the selection in the **Options** field. If the option is **Create Table**, then the default target value is the name of source table. To specify a different name for the target, enter it in the **Name** field. For the other options, the default value is <None>. Expand the drop-down list and select a table as the target.

In the **Options** field for the source, select **Create Table**, **Insert into Table**, **Replace Data**, **Drop Table and Create New Table**, or **Merge into Table**. Point to the question mark icon to see a brief description of the selected option.

Select a different schema from the **Schema** drop-down to create your target table in another schema.

> **Note:**
>
> The **Schema** drop-down is available only if you have `PDB_DBA` role grant to you.
>
> To grant yourself a `PDB_DBA` role, you must log into your Database Actions instance and enter the following command in the **SQL** worksheet area displayed in the **SQL** tab under **Development** tools present in the Launchpad.
>
> ```
> Grant PDB_DBA to Username;
> ```
>
> This drop-down is available for **Create Table** and **Drop Table and Create New Table** options.

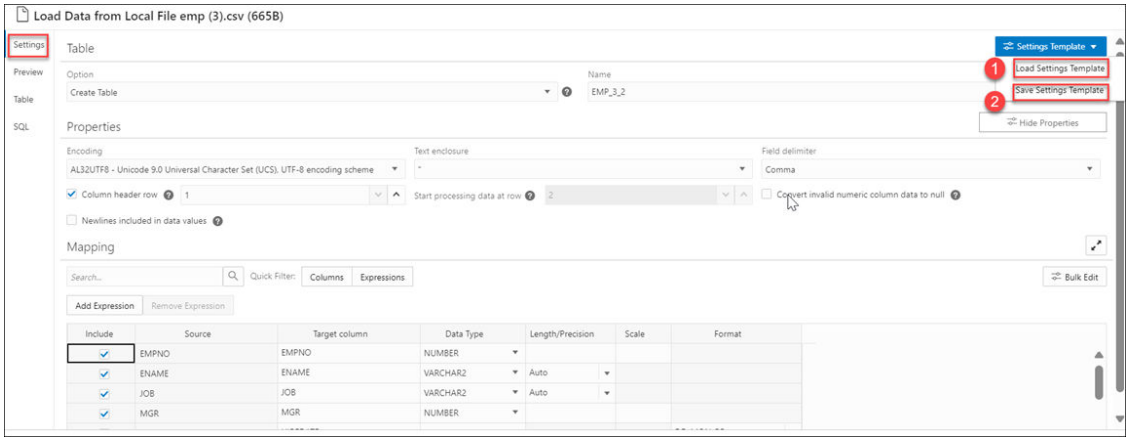
If you select **Create Table**, then in the **Name** field accept the default name, which is the name of the source table, or enter a different name.

If you select one of the other options, then expand the drop-down list of the **Name** field and select a table as the target.

Settings Template

The save settings feature saves the configuration set in the Cart settings in the form of a JSON file. When opening the Settings template, you have the following options:

1. **Load Settings Template**: Loads a settings template from your local system.
2. **Save Settings Template**: Saves the current existing settings template.



You can use the **Load Settings Template** if you want to use an existing customized template present in your local.

1. From the Settings Template in the Settings tab of the Load Data page, select **Load Settings Template**.
2. You will see a Load Settings Template wizard, click the Settings Template JSON to load a JSON file from your system.
3. Clicking the Settings template JSON will open your local system. Click **OK** to load the JSON file.
4. After you load the JSON file, you can view the updates applied automatically to the settings tab which matches the JSON settings template you load from your local.

You can use the **Save Settings Template** to save the existing current Settings template.

1. From the Settings Template in the Settings tab of the Load Data page, select **Save Settings Template**.
2. The Template file editor appears. Click the Template File name and name the new template.
3. Click **OK** to finish saving the new name of the existing template.
4. You can test the configuration of the new template.

Bulk Edit Settings

You can use the Bulk edit settings to update all the columns at once from the mapping table. Use it to apply changes to the selection currently displayed in the results pane. You can search for the values of the column you want to edit in the search field and click the magnifier icon. The mapping table will display the results of the search. Select the Bulk Edit setting to update the column. The Bulk Edit setting allows you to:

- Update values of all the fields in a group.
- Find and replace, Add Prefix and Add suffix to target column name.
- Include the column(s) for loading data to the target table.
- Exclude the column(s) for loading data to the target table.

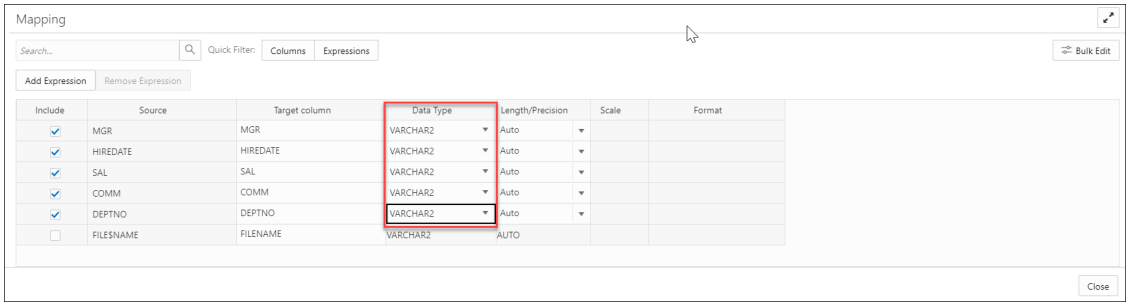
Searching the Column

The Bulk Edit setting updates the columns returned by the search field. The search box besides the Bulk Edit setting icon filters the list of columns you wish to update in a bulk. As soon as you start typing in the search field, the tool returns the field values which contains the letters you type. You can remove the filter by deleting all the content from the search box and clicking the magnifier icon that appears next to the search box.

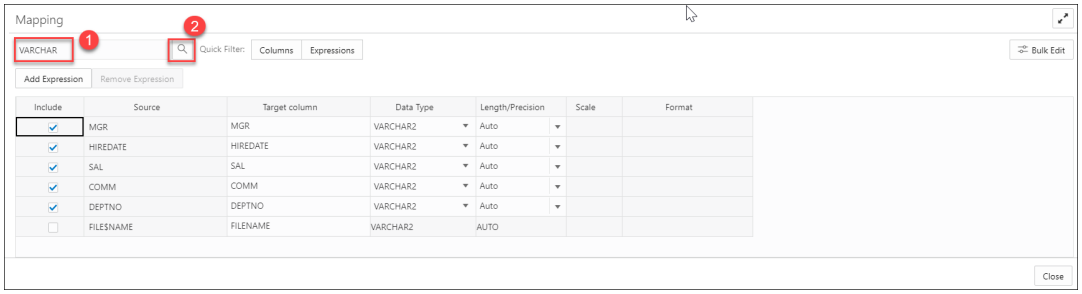
The Bulk Edit setting enables you to update the values of the following columns for all the searches returned by the search field:

- Data Type
- Target Column name
- Include Columns for loading
- Exclude columns for loading

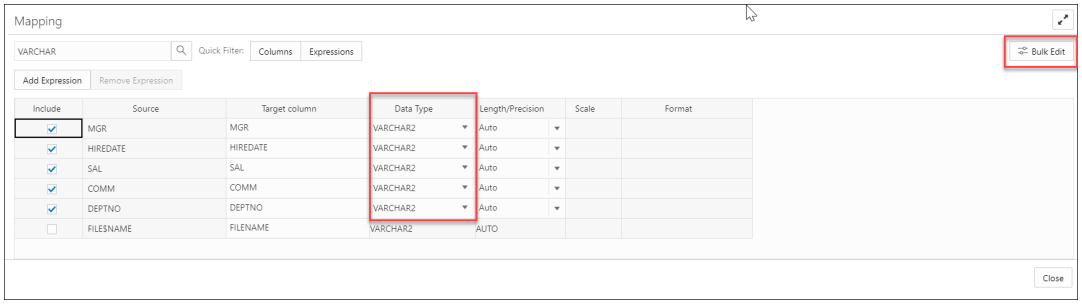
Consider changing the Data Type of first five rows from VARCHAR to NUMBER in the mapping table.



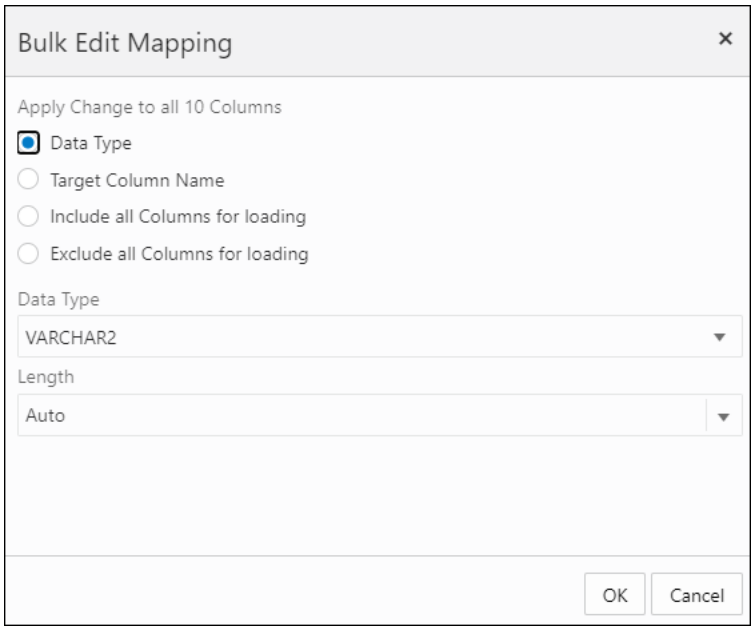
- Type `VARCHAR` in the search field and click on the magnifier icon besides the search field.



- The search will return the rows having `VARCHAR` as the Data Type. Select the **Bulk Edit** icon besides the magnifier icon.

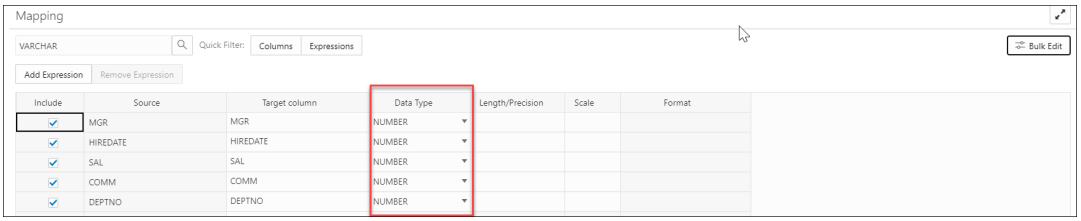


- Selecting the Bulk Edit icon opens the Bulk Edit Mapping dialog box.

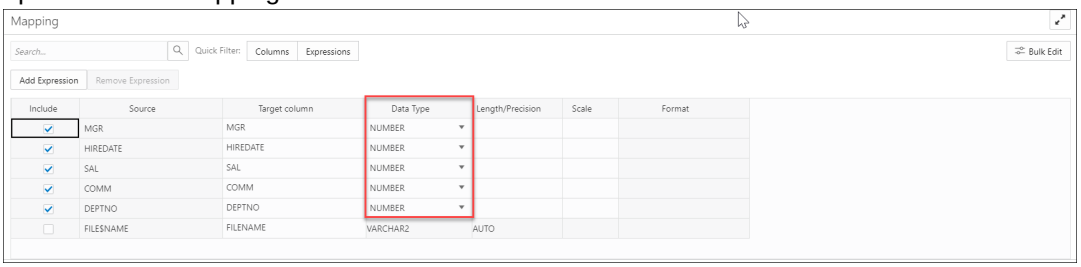


- Click **Data Type** and select `NUMBER` from the Data Type drop-down. Click **OK**.

- The Data Type of the selected rows changes from `VARCHAR` to `NUMBER`.



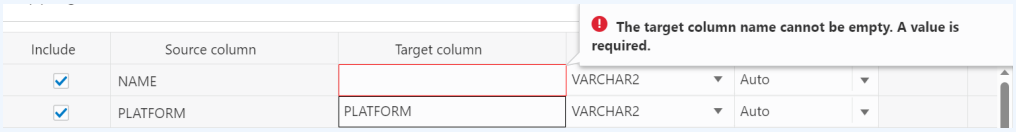
- Clear the content from the search field and click the magnifier icon to view the bulk edit updates in the Mapping table.



Specify Mappings

If you select the **Create Table** or the **Drop Table and Create New Table** option, then in the **Mapping** section either accept the default values for the target columns or specify different values. For the target column, enter a name for the column.

> **Note:**
>
> You will receive a tooltip error message with the exact reason for the error message when you complete editing a cell. The mapping grid cell will be highlighted with red to indicate an invalid value that must be fixed. The highlight is removed after you fix the invalid value. For example, you can view the following tooltip error message when the target column name is not filled in.



For the **Insert into Table** or **Replace Data** options, select a target column from the drop-down list of existing columns.

For the **Merge into Table** option, for each source column, select a target column form the drop-down list. You must specify at least one column as a key column. To specify a column as a key column, select the **Merge Key** check box for the column. Merge keys are one or more columns that uniquely identify each row in the table. Merge keys must not contain any null values. For loading tables with primary keys, this option automatically enables the selection of primary key columns as the merge keys.

Preview

To view the data in the source table, in the settings pane select the **Source Table** tab. The source preview displays the data in the table.

Table

For all options except **Create Table**, to view the existing data in the target table, in the settings pane select the **Target Table** tab. The target preview displays the data in the table before you run the data load job.

SQL

The SQL tab displays the SQL commands that will be run to complete this data load job.

> **Note:**
>
> You can see the SQL code even before the table is created.

Run the Data Load Job

When you have added all of the source tables for the job and specified the settings for each table, to run the job click the Start icon in the Data Load Cart menu bar. To stop the data load job, click the Stop icon.

Once the data load job starts, you can view the progress of the job in the Data Load dashboard.

When the data load job completes, the Data Load dashboard page displays the results of the job. At the top of the header of the table load, you can view the name of the table along with the total columns present in the table.

Click **Report** to view a report of the total rows loaded and failed for a specific table.You can view the name of the table, the time the table was loaded and the time taken to process the load.

At the header of the Table Load, you can view the name of the table with total number of columns loaded. When you expand the Table Load you can view the options you used to load the source data with the count of rows loaded.

To view information about an item in the job, click the Settings icon in the item. The settings pane has the same **Settings**, **Source**, **Table**, **SQL**, **Job Report** and **Data Definition** tabs as the settings pane before running the job, except that the target preview now contains the data loaded by the data load job. To close the settings pane, click **Close**.

To view a log of the load operation, click the Log icon. You can save the log, clear it, or refresh it. Click **OK** to dismiss the log.

The list of tables on the Data Load / Explore page contains any new tables created. The target tables for the **Insert into Table**, **Replace Data**, **Drop Table and Create New Table**, and **Merge into Table** options contain the loaded data.

Fixing a data load job

After your data load job, you might see errors that you want to correct, or upon inspection, realize that you wanted to name a column differently. In such cases, you will view a warning sign on the Table load. Click the **Reload** icon to reload source with suggested fixes. Click **Actions** icon on the Table load and select **Edit** to make any changes to the data load job (i.e., change a column name).

Click **Apply** to apply any changes. Click **Close** to return to the Database Actions page.

Loading Data from Cloud Storage

You can load data from a cloud store to a table in your Autonomous Database.

You can load files in these file formats: AVRO, CSV, JSON, GeoJSON, ORC, Delimited TXT, XLSX, PRQ, GZ, GNU ZIP and Tab-Separated Values. For information on supported file formats, see Format Specifications for JSON and AVRO Files. The Data Load tool supports loading tables from multiple worksheets XLSX files when the file is in a cloud store.

You can set filters on the data for a table to load only the specified data. For example, to limit the files to only those that are CSV files, enter *.CSV in the file extension filter.

Configure and run a data load job from the Load Cloud Object page. To open that page:

1. Open the Database Actions and select **Data Load**.
2. Select **LOAD DATA** and **Cloud Store** .

On the left side of the page is a *navigator pane*, where you choose a cloud store connection and the folders or files containing the data. On the right of the page is the data load *cart*, where you stage the files and folders for the data load job. You can set options for the data load job before running it. The Autonomous Database comes with predefined CPU/IO shares assigned to different consumer groups. You can set the consumer group to either low, medium or high while executing a data load job depending on your workload.

To load files from a cloud store into your database, do the following:

- Manage Cloud Storage Links for Data Load Jobs
- Prepare the Data Load Job
- Add Files or Folders for the Data Load Job

- Enter Details for the Data Load Job
- Run the Data Load Job
- View Details About the Data Load Job After It Is Run
- View the Table Resulting from the Data Load Job
- Manage Cloud Storage Links for Data Load Jobs
  Before you can load data from a cloud store, you must establish a connection to the cloud store you want to use. You can select cloud store location from the cloud store locations field.
- Prepare the Data Load Job
- Add Files or Folders for the Data Load Job
- Enter Details for the Data Load Job
  Enter the details about the data load job in the Load Data from Cloud Storage pane.
- Run the Data Load Job
  Once you've added data sources to the data load cart and entered details about the data load job, you can run the job.
- View Details About the Data Load Job After It Is Run
- View the Table Resulting from the Data Load Job

Manage Cloud Storage Links for Data Load Jobs

Before you can load data from a cloud store, you must establish a connection to the cloud store you want to use. You can select cloud store location from the cloud store locations field.

On the Load Data page when you select Cloud Store:

1. Click the **Create Cloud Store locations** menu besides the cloud store locations text field. This opens an Add Cloud Store Location dialog box. See Managing Connections to add cloud store location.

See Managing Connections.

To return to the Load Cloud Object page, click **Data Load** in the breadcrumbs at the top of the page and then navigate back to the page.

Prepare the Data Load Job

As you'll see below in Enter Details for the Data Load Job, the first decision you'll make when configuring a data load job is how to load the source data into a new or existing table in the database. The choices are:

- Create a table and insert data loaded from the source into the new table.
- Insert data loaded from the source into an existing table.
- Delete all data in an existing table and then insert new data from the source into the table.
- Drop a table, create a new table, and then insert data loaded from the source into the new table.
- Merge data from the source into a table by updating existing rows in the table and inserting new rows into the table.

You may have to adjust your source data or your target table so that the source data loads correctly into the external target table. The number, order, and data types of columns in the source must match those in the target. Consider:

- If you're creating a new table or if the columns in your source exactly match the columns in an existing target, you don't have to do any special preparation.

- If the columns in your source don't match the columns in an existing target, you must edit your source files or target table so they do match.
- If you're loading multiple files, you must make sure that:
  - All the source files are of the same type, for example, CSV, JSON, etc.
  - The number, order, and data types of the columns in all the source files match (and that they match the target, if you're loading into an existing table).
- If you want to partition by date:
  - The source file must contain data where the data type is date or timestamp.
  - You must load a folder containing two or more data sources.
  - The names of the files in the folder must indicate a date or dates, for example, `MAR-1999.csv` or `2017-04-21.xlsx`.

Add Files or Folders for the Data Load Job

Add files from the cloud store to the data load cart, where you can edit the details of the data load job. To add the files:

1. From the list at the top of the navigator pane on the left, select the bucket from the drop-down with your source data.

   The list shows links that were established on the Manage Cloud Storage page. If you haven't yet registered the cloud store you want to use, click the **Create Cloud Store Location** button at the top of the page and register a connection. See Manage Cloud Storage Links for Data Load Jobs, above.

2. Drag one or more items from the file navigator on the left and drop them into the cart on the right.
   - You can add files, folders, or both. A card is added to the cart for each file or folder you drag into it. The card lists the name of the source file or folder and a proposed name for the target table.
   - If you add a folder that contains multiple files, all the files must be of the same type, that is, CSV, TXT, etc.

     When you add the folder to the cart, a prompt is displayed that asks if you want to load all the objects from the multiple source files into a single target table. Click **Yes** to continue or **No** to cancel.
   - When you add multiple individual files or multiple folders to the cart, the data represented by each card will be loaded into a separate table, but all the items in the cart will be processed as part of the same data load job. The multiple files you load must have same file extension.
   - You can add files or folders from a different bucket, but if you do that, you're prompted to remove all files that are already in the cart before proceeding. To select files from a different bucket, select the bucket from the drop-down list in the navigator pane on the left and then add the file(s), as described above.
   - You can drop files or folders into the data load cart and then navigate away from the Data Load Object page. When you return to the page, those items remain on the page.

You can remove items from the cart before running the data load job:

- To remove an item from the cart, click the **Remove** icon on the card for the item.
- To remove all items from the cart, click **Remove All** in the data link cart menu bar at the top of the pane besides the **Start** and **Stop** icon.

Enter Details for the Data Load Job

Enter the details about the data load job in the Load Data from Cloud Storage pane.

On the card in the data link cart, click the **Actions** icon and select the **Settings** to open the Load Data from Cloud Storage pane for that job. The pane contains:

- Settings Tab - Table Section
- Settings Tab - Properties Section
- Settings Tab - Mapping Section
- Preview Tab
- Table Tab
- Close Button - Save and Close the Pane

Settings Tab - Table Section

Set details about the target table in the **Table** section.

- **Option:** Select an item from the **Option** list to specify how the data should be loaded into a new or existing table. The processing options are:
  - **Create Table**: Creates a table and inserts the data into the new table. When you select this option, the **Name** field on the **Settings** tab is filled with a default name, based on the name of the source file or folder. You can change it if you want.
  - **Insert into Table**: Inserts data loaded from the source into an existing table. When you select this option, the **Name** field on the **Settings** tab presents a list of the tables in the current schema. Select the table into which you want to insert the data.
  - **Replace Data**: Deletes all data in the existing table and then inserts new data from the source into the table. When you select this option, the **Name** field on the **Settings** tab presents a list of the tables in the current schema. Select the table you want to use.
  - **Drop Table and Create New Table**: Drops the table (if it already exists), creates a new table, and then inserts the new data into the table. When you select this option, the **Name** field on the **Settings** tab presents a list of the tables in the current schema. Select the table you want to use.
  - **Merge into Table**: Updates existing rows and inserts new rows in the table. When you select this option, the **Name** field on the **Settings** tab presents a list of the tables in the current schema. Select the table you want to use.
- Select a different schema from the **Schema** drop-down to create your target table in another schema.

> **Note:**
>
> The **Schema** drop-down is available only if you have `PDB_DBA` role grant to you.
>
> To grant yourself a `PDB_DBA` role, you must log into your Database Actions instance and enter the following command in the **SQL** worksheet area displayed in the **SQL** tab under **Development** tools present in the Launchpad:
>
> ```
> Grant PDB_DBA to USER;
> ```
>
> You can now view the **Schema** drop-down. It is only available for **Create Table** and **Drop Table and Create New Table** options.

- **Name:** The name of the target table.
- **Partition Column:**

  List Partitions and Date-based partitions are the different types of partitions available in data loading.

  List partitioning is required when you specifically want to map rows to partitions based on discrete values.

  To partition according to a specific column, click the **Partition Column** drop-down list and select the column you want to use for the partitioning.

  You will have N files per partition value, all partitioned by the partition column you select.

  > **Note:**
  >
  > – For linked files (from external tables) there is also a requirement that for each file, the list partitioning column can contain only a single distinct value across all of the rows.
  > – If a file is list partitioned, the partitioning key can only consist of a single column of the table.

  Date-based partitioning is available when you load a folder containing two or more data sources that contain date or timestamp data.

  To partition according to date, click the **Partition Column** drop-down list and select the **DATE** or **TIMESTAMP** column you want to use for the partitioning.

Settings Tab - Properties Section

Specify options to control how the source data is interpreted, previewed, and processed. These options vary, depending on the type of source data.

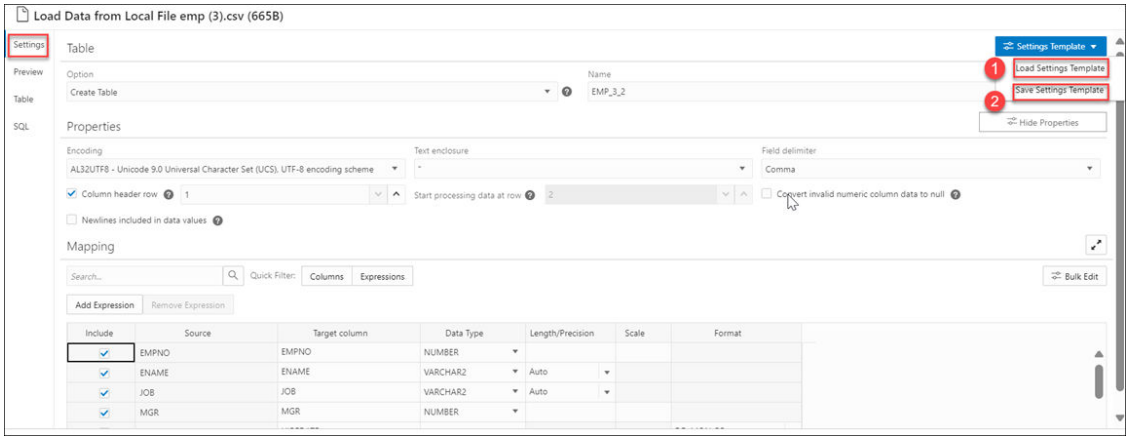
- **Encoding:** Select a character encoding type from the list. This option is available when the loaded file is in plain text format (CSV, TSV, or TXT). The default encoding type is UTF-8.
- **Text enclosure:** Select the character for enclosing text: **"** (double-quote character), **'** (single-quote character) or **None**. This option is visible only when the selected file is in plain text format (CSV, TSV, or TXT).

- **Field delimiter:** Select the delimiter character used to separate columns in the source. For example, if the source file uses semicolons to delimit the columns, select **Semicolon** from this list. The default is **Comma**. This option is visible only when the selected file is in plain text format (CSV, TSV, or TXT).
- **Column header row**: Select the **Column header row** checkbox to use the column names form the source table in the target table.
  - By selecting this option you can indicate what row in the file contains column names. The rows in the **Mapping** section, below, are filled with those names (and with the existing data types, unless you change them).
  - If you deselect this option, the first row is processed as data. To specify column names manually, enter a name for each target column in the **Mapping** section. (You will also have to enter data types.)
- **Start processing data at row**: Specifies the number of rows to skip when loading the source data into the target:
  - If you select the **Column header row** option under **Source column name** (see below) and if you enter a number greater than 0 in the **Start processing data at row** field, then that number of rows after the first row are not loaded into the target.
  - If you deselect the **Column header row** option under **Source column name**, and if you enter a number greater than 0 in the **Start processing data at row** field, then that number of rows including the first row are not loaded into the target.
- **Numeric column**: Select the **Convert invalid data to null** checkbox to convert an invalid numeric column value into a null value.
- **Newlines included in data values**: Select this option if there are newline characters or returns to the beginning of the current line without advancing downward in the data fields. Selecting this option will increase the time taken to process the load. If you do not select this option when loading the data, the rows with newlines in the fields will be rejected. You can view the rejected row in the Job Report panel.

Settings Template

The save settings feature saves the configuration set in the Cart settings in the form of a JSON file. When opening the Settings template, you have the following options:

1. **Load Settings Template**: Loads a settings template from your local system.
2. **Save Settings Template**: Saves the current existing settings template.



You can use the **Load Settings Template** if you want to use an existing customized template present in your local.

1. From the Settings Template in the Settings tab of the Load Data page, select **Load Settings Template**.
2. You will see a Load Settings Template wizard, click the Settings Template JSON to load a JSON file from your system.
3. Clicking the Settings template JSON will open your local system. Click **OK** to load the JSON file.
4. After you load the JSON file, you can view the updates applied automatically to the settings tab which matches the JSON settings template you load from your local.

You can use the **Save Settings Template** to save the existing current Settings template.

1. From the Settings Template in the Settings tab of the Load Data page, select **Save Settings Template**.
2. The Template file editor appears. Click the Template File name and name the new template.
3. Click **OK** to finish saving the new name of the existing template.
4. You can test the configuration of the new template.

Bulk Edit Settings

You can use the Bulk edit settings to update all the columns at once from the mapping table. Use it to apply changes to the selection currently displayed in the results pane. You can search for the values of the column you want to edit in the search field and click the magnifier icon. The mapping table will display the results of the search. Select the Bulk Edit setting to update the column. The Bulk Edit setting allows you to:

- Update values of all the fields in a group.
- Find and replace, Add Prefix and Add suffix to target column name.
- Include the column(s) for loading data to the target table.
- Exclude the column(s) for loading data to the target table.

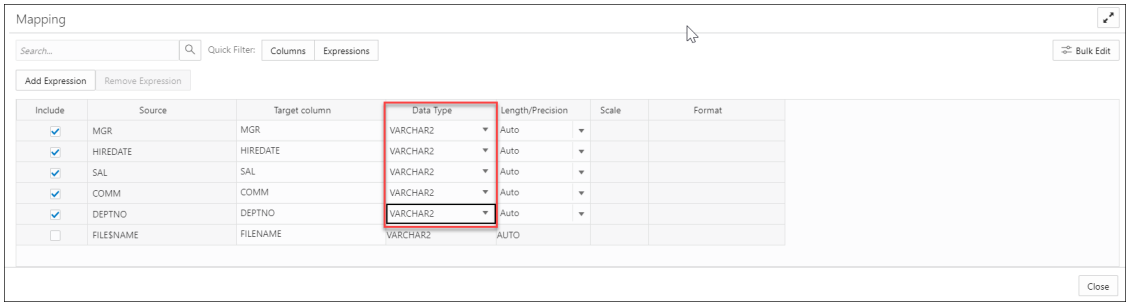
Searching the Column

The Bulk Edit setting updates the columns returned by the search field. The search box besides the Bulk Edit setting icon filters the list of columns you wish to update in a bulk. As soon as you start typing in the search field, the tool returns the field values which contains the letters you type. You can remove the filter by deleting all the content from the search box and clicking the magnifier icon that appears next to the search box.

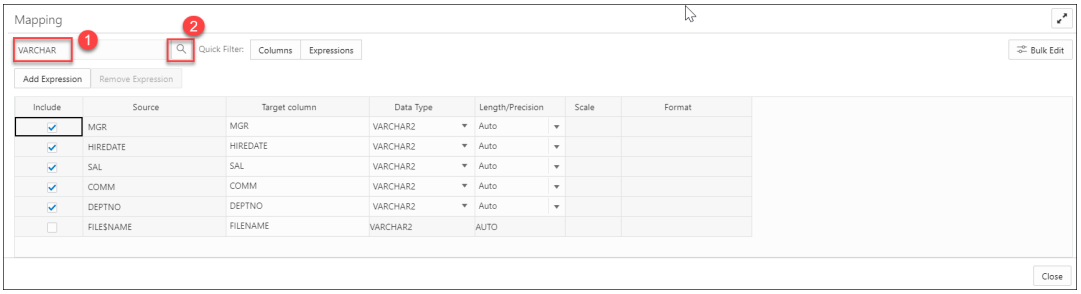
The Bulk Edit setting enables you to update the values of the following columns for all the searches returned by the search field:

- Data Type
- Target Column name
- Include Columns for loading
- Exclude columns for loading

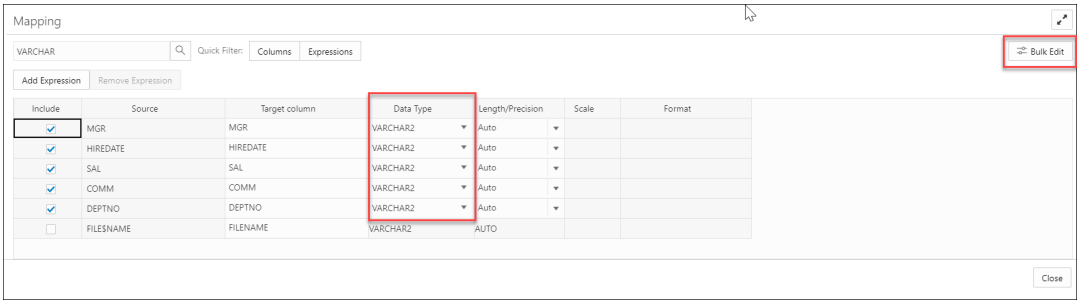
Consider changing the Data Type of first five rows from VARCHAR to NUMBER in the mapping table.



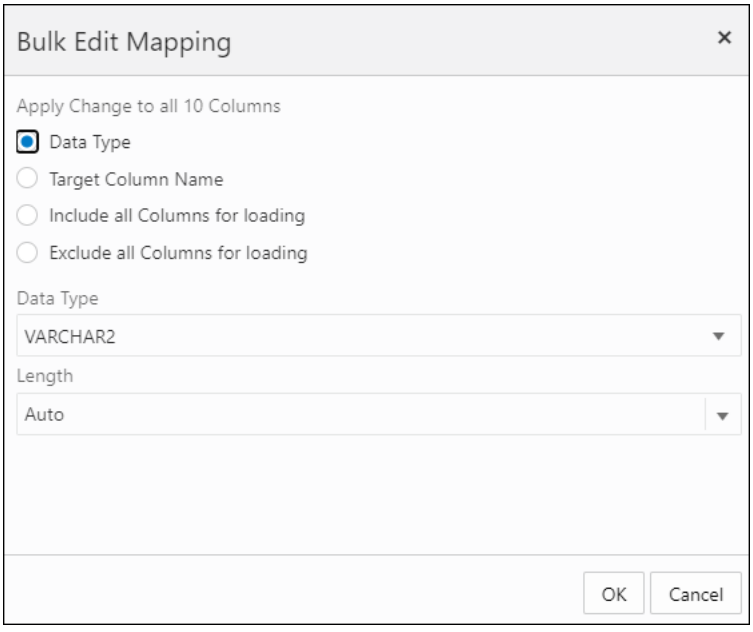
- Type `VARCHAR` in the search field and click on the magnifier icon besides the search field.



- The search will return the rows having `VARCHAR` as the Data Type. Select the **Bulk Edit** icon besides the magnifier icon.

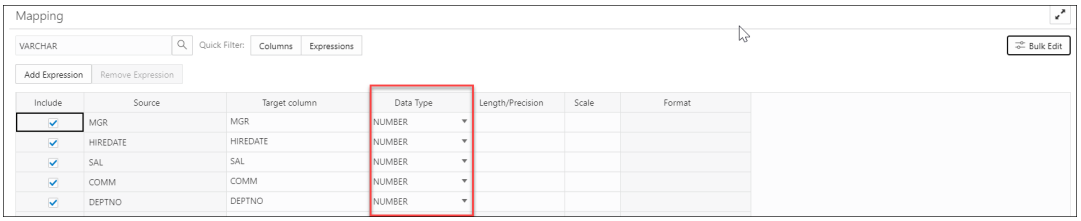


- Selecting the Bulk Edit icon opens the Bulk Edit Mapping dialog box.

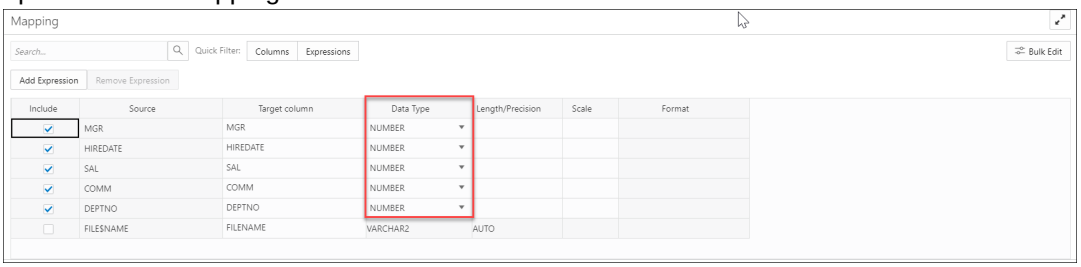


- Click **Data Type** and select `NUMBER` from the Data Type drop-down. Click **OK**.

- The Data Type of the selected rows changes from `VARCHAR` to `NUMBER`.



- Clear the content from the search field and click the magnifier icon to view the bulk edit updates in the Mapping table.



Settings Tab - Mapping Section

The settings in the **Mapping** section control how data from the source files are loaded into the rows of the target database table. For each row, the data from the column listed under **Source column** will be loaded into the column listed under **Target column**.

As mentioned above, the contents of the **Mapping** table change according to what processing option you chose in the **Table** section and which properties you set in the **Properties** section.

You can filter the results in the mapping table with **Quick Filter** field that enables you to filter out **Columns** or **Expressions**.

Select **Add Expression** to perform **Sentiment Analysis** or **Key Phrase extraction** with the source data. See Use OCI Language Service Capabilities in Data Studio for more details.

- **Include**: This check box ensures that the row you select is loaded from the source column to the target column.

- **Source column**: Lists the columns from the source file.

  If the **Column header row** option under **Properties** is selected, **Source column** shows the names of the columns in the source file. If the **Column header row** option is not selected, generic names like **COLUMN_1**, **COLUMN_2**, etc., are used. This field is always read only.

  You can view two source columns `FILE$NAME` and `SYSTIMESTAMP`. The `FILE$NAME` column enables you to locate the source file containing a particular data record. For example, you load a source file that contains a list of files. The file names in the file list refer to the department names across the organization. For instance, a *finance.txt* file contains data from the Finance department. In the mapping, you can use string data types to extract the department name from the output of the file name column. You can use the extracted department name to process data differently for each department.

  The `SYSTIMESTAMP` column allows us to view the current timestamp in the database.

  > **Note:**
  >
  > – `FILE$NAME` and `SYSTIMESTAMP` source columns are not included by default. You must check the **Include** check box and run the load for the target table to display these two columns.
  >
  > – When you are creating a livefeed, the `FILE$NAME` and `SYSTIMESTAMP` source columns appear in the Mapping table by default.

- **Target column:** Lists the columns in the target table. Accept, select, or enter a column in the target table. You need to make sure that the target column is not empty. The target column name cannot have duplicate name as another target column. The target column length must not be beyond 128 bytes. 128 byte limit is a database limit.

  The contents of this column differ, depending on what you selected for the table processing **Option** and whether you selected for the **Column header row** option.

  – If (1) the processing option is **Create Table** or **Drop Table and Create New Table** and (2) the **Column header row** option *is* selected, then the **Target column** uses the names of the columns in the source file. You can change the name of a target column by replacing the provided name with a new one.

  – If (1) the processing option is **Create Table** or **Drop Table and Create New Table** and (2) the **Column header row** option is *not* selected, then generic names like **COLUMN_1**, **COLUMN_2**, etc., are used. You can change the name of a target column by replacing the provided name with a new one.

  – If (1) the processing option is **Insert into Table**, **Replace Data**, or **Merge Into Table** and (2) the **Column header row** option *is* selected, then the **Target column** has a drop-down list of all the columns in the target table, with their data types. By default, the column with the name corresponding to the source column is selected, but you can select a different one from the list.

  – If (1) the processing option is **Insert into Table**, **Replace Data**, or **Merge Into Table** and (2) the **Column header row** option is *not* selected, then the **Target column** has a drop-down list of all the columns in the target table, with their data types. Select a column from the list to use as the target column.

> **Note:**
>
> If you're loading multiple files from a folder in a single data load job, only the first file will be shown in the **Mapping** section. However, as long as the column names and data types match, the data from all source files will be loaded.

- **Data Type:** Lists the data type to use for data in that column. This column is displayed only for **Create Table** or **Drop Table and Create New Table**. The contents change depending on whether the **Get from file header** option is selected.
  - If the **Get from file header** option *is* selected, **Data type** shows the data types of the columns in the source file (for **Create Table**) or in the existing table (for **Drop Table and Create New Table**). If you want to change the data type for the target, click the name and select a different one from the list.
  - If the **Column header row** option is *not* selected, **Data type** shows all available data types. Select the data type to use for the target column from the list.
- **Length/Precision** (Optional): For columns where the **Data Type** is **NUMBER**, enter the length/precision for the numbers in the column. Precision is the number of significant digits in a number. Precision can range from 1 to 38.

  For columns where Data Type is VARCHAR2, the **Auto** value in **Length/Precision** field enables the Auto Size feature.

  With the Auto-Size column Width feature, you can automatically size any column to fit the largest value in the column. Select **Auto** from the **Length/Precision** drop-down values or pick a value from the drop-down list.
- **Scale** (Optional): For columns where the **Data Type** is **NUMBER**, enter the scale for the numbers in the column. Scale is the number of digits to the right (positive) or left (negative) of the decimal point. Scale can range from ranges from -84 to 127.
- **Format:** If the data type in the **Data type** column is **DATE** or one of the **TIMESTAMP** types, select a format for that type from the from the **Format** drop-down list.
- **Merge Key:** This option is used only for the processing option **Merge into Table**.

  For the **Merge into Table** option, you must specify at least one column to use as a key column. Merge keys are one or more columns that uniquely identify each row in the table. To specify a key column, select the **Merge Key** checkbox for the column. Merge keys must not contain any null values. For loading tables with primary keys, this option automatically enables the selection of primary key columns as the merge keys.

Preview Tab

The **Preview** tab displays the source data in tabular form in the **Load Preview** menu. The display reflects the settings you chose in the **Properties** section.

The **File** menu in the **Preview** tab displays the source data with all the columns with their names. You can copy the text.

Table Tab

The **Table** tab displays what the target table is expected to look like after the data has been loaded. If you chose the **Create Table** processing option, no table is shown.

SQL Tab

The **SQL** tab displays the SQL commands that will be run to complete this data load job.

> **Note:**
>
> You can see the SQL code even before the table is created.

Close Button - Save and Close the Pane

After entering all the details for the data load job, click **Close** at the bottom of the page. This saves the details you entered and returns you to the Load Data from Cloud Storage pane. To close the page without saving your entries, press **Escape**.

Run the Data Load Job

Once you've added data sources to the data load cart and entered details about the data load job, you can run the job.

To run the job:

1. If you haven't already done so, click the **Close** button in the **Load Data from Cloud Storage** pane to save your settings and close the pane. If any of the settings are invalid, an error message reports the problem. Fix the problem and click **Close**.

2. Click **Start** in the data load cart menu bar. To stop the data load job, click **Stop**.

   When you have added all of the sources for the job and specified the settings for each source, to run the job click the Start icon in the Data Load Cart menu bar. In the Run Data Load Job dialog box, click **Start**. To stop the data load job, click the Stop icon.

   Once the data load job starts, you can view the progress of the job in the Data Load dashboard.

   When the data load job completes, the Data Load dashboard page displays the results of the job. At the top of the header of the table load, you can view the name of the table along with the total columns present in the table.

   Click **Job Report** to view the total number of rows processed successfully and the count of rejected rows. You can also view the Start time. The SQL pane of the Job Report displays the equivalent SQL code of the job.

   To view information about an item in the job, click the **Actions** icon on the Table Load.

   To view a log of the load operation, click the Logging icon. You can save the log, clear it, or refresh it. Click **OK** to dismiss the log.

View Details About the Data Load Job After It Is Run

To view details about the data load job after it is run, click the **Actions** icon on the **Table and View Loads** section of the Data Load dashboard.

You can view Table details, Gather Statistics, Register to Cloud Link, Create Analytic View, Export Data to Cloud, Edit Table and delete the table.

When the data load job completes, the Data Load dashboard page displays the results of the job. At the top of the header of the table load, you can view the name of the table along with the total columns present in the table.

Click **Report** to view the total number of rows processed successfully and the count of rejected rows. You can also view the Start time. The SQL pane of the Report displays the equivalent SQL code of the job.

After your data load job, you might see errors that you want to correct, or upon inspection, realize that you wanted to name a column differently. In such cases, you will view a warning

sign on the Table load. Click the **Reload** icon to reload source with suggested fixes. Click **Actions** icon on the Table load and select **Edit** to make any changes to the data load job (i.e., change a column name).

View the Table Resulting from the Data Load Job

To view the new tables or tables modified by the data load job, you can:

1. Fix your data load job. After your data load job, you might see errors that you want to correct, or upon inspection, realize that you wanted to name a column differently. In such cases, click the **Reload** option on the selected Table Load to reload cards from your recent cart and edit them as you did before your first attempt. The Reload icon reloads the source data with the fixes suggested by the tool. Click the **Actions** icon on the Table header, click **Table** and select **Edit** to make any changes to the data load job (i.e., change a column name).
2. You can use the **Edit** icon as mentioned in the above paragraph to review the table.

Linking Data

You can link to data in remote databases or in cloud storage buckets.

When you link to data in a remote database or in cloud storage, the target object produced is an external table or a view. When you select that target table or view on the Data Load Jobs page, the source preview for the object shows the current data in the source object.

Linking to columns in a remote database or to files in cloud storage can be useful when the source data is being continually updated. For example, if you have a database table with columns that are updated for each new sales transaction, and you have run a data load job that links columns of the source table to targets in your Oracle Autonomous Database, then those targets have the new sales data as it is updated in the source.

See:

- Linking to Other Databases
- Linking to Objects in Cloud Storage
- Linking to Directory
- Linking to Share
- Linking to OCI Data Catalog

- Linking to Other Databases
  To link to data in tables in another database from your Oracle Autonomous Database, on the Data Load Dashboard, click **LINK DATA**, then click **Database**. Select a database link from the drop-down list. Drag one or more tables from the list of database tables and drop them in the Data Load Cart.
- Linking to Objects in Cloud Storage
  When you create a link to files in a cloud store bucket from you Oracle Autonomous database, you create an external table that links to the files in the cloud store.
- Linking to File System
  You can link to file system directories from your Autonomous Database.
- Linking to Share
  You can subscribe to a Share Provider for linking data.
- Linking to Data Catalog
  You can link your Data Catalogs to Autonomous Database.

Linking to File System

You can link to file system directories from your Autonomous Database.

On the Data Load dashboard page, click **Link Data** and select **File System**. On the left side of the page is a navigator pane, where you choose a directory with the folders or files containing the data. On the right of the page is the data link cart, where you stage the files and folders for the data link job. You can set options for the data link job before running it. The Autonomous Database comes with predefined CPU/IO shares assigned to different consumer groups. You can set the consumer group to either low, medium or high while executing a data load job depending on your workload.

Drag and drop the files or folders from the directory you select to add them to linking job.

Enter Details for the Data Link Job

Enter the details about the data link job in the Link Data from Directory dialog box. For more details, refer to the Linking to Objects in Cloud Storage chapter.

Linking to Share

You can subscribe to a Share Provider for linking data.

On the Data Load Dashboard page, click **Link Data** and select **Share**. For more details, refer to the Loading data from a Share chapter.

Linking to Data Catalog

You can link your Data Catalogs to Autonomous Database.

Click **OCI Data Catalog** in the Link Data page under the Data Load menu to register a Data Catalog for linking data. Refer to the Register Data Catalog chapter for more details on this.

Linking to Other Databases

To link to data in tables in another database from your Oracle Autonomous Database, on the Data Load Dashboard, click **LINK DATA**, then click **Database**. Select a database link from the drop-down list. Drag one or more tables from the list of database tables and drop them in the Data Load Cart.

Each table appears as an item in the Data Load Cart. The item shows the name of the table and the number of rows in it, and the name of the table that is the target for the data load.

To remove a table from the Data Load Cart, select the Remove (trash can) icon for the item. To remove all tables from the cart, click the Remove All (trash can) icon in the Data Load Cart menu bar.

To add a remote database to the list of databases, create a database link to the remote database. For information on creating a database link, see Database Links in *Oracle® Database Database Administrator's Guide*.

The databases available to you appear in the drop-down list of the database navigation pane of the Link Tables page.

Specify the Target, Create Filters, and View Mappings

To specify settings for the data load job, preview the data in the source or the target, and see statistics about the data, click the Settings (pencil) icon for the item in the Data Load Cart.

In the settings pane, on the Settings tab, the **Source** field displays the name of the table and the number of rows in the table.

Preview Source Data

To view the data in the source table, in the settings pane select the **Source Preview** tab. The source preview displays the data in the table.

View Statistics

To view statistics about the source table, in the settings pane select the **Statistics** tab. It may take a moment for the statistics to appear. The statistics include the size of the table, the number of rows and columns, the column names, data types, number of distinct values, and other information. Below the details about the columns is a bar graph that displays the top unique values for the selected column.

Preview Target Data

To view the data in the target view, in the settings pane select the **Target Preview** tab. The target preview displays the data in the target view. If the view does not yet exist, then the target data is the same as the source data, regardless of any filters set for the data load job.

Run the Data Load Job

When you have added all of the source tables for the job and specified the settings for each table, to run the job click the Start icon in the Data Load Cart menu bar. In the Run Data Load Job dialog box, click **Start**. To stop the data load job, click the Stop icon.

At the top of the page, the **Status** shows the number of items for which the load has completed over the number of items in the job, and the total time elapsed for the job. When the data load job completes, the Link Tables page displays the results of the job.

To view information about an item in the job, click the Settings (circled i) icon in the item. The settings pane has the same **Settings**, **Preview**, and **Target Preview** tabs as the settings pane before running the job, except that the **Target Preview** now contains the data loaded by the data load job. To close the settings pane, click **Close**.

To view a log of the load operation, click the Logging icon. You can save the log, clear it, or refresh it. Click **OK** to dismiss the log.

The list of views on the Data Load page contains any new views created. A preexisting view that was the target for the data load job now contains the loaded data.

On the Database links page, click the **Reload Cart** button. Clicking the button enables you to add the data sources to the data load the cart again.

Click **Done** to return to the Database Actions page.

Linking to Objects in Cloud Storage

When you create a link to files in a cloud store bucket from you Oracle Autonomous database, you create an external table that links to the files in the cloud store.

You can link to files in these file formats: AVRO, CSV, JSON, GeoJSON, Parquet, ORC, Delimited TXT. For information on supported file formats, see Format Specifications for JSON, AVRO, and XML Files.

Configure and run a data link job from the Link Cloud Object page. To open that page:

1. Open the Database Actions launchpad , click **Data Studio** tab and select the **Data Load** menu. See, The Data Load Page.

2. Select **LINK DATA** and **CLOUD STORE**.

On the left side of the page is a *navigator pane*, where you choose a cloud store connection and the folders or files containing the data. On the right of the page is the data load *cart*, where you stage the files and folders for the data link job. You can set options for the data link job before running it. The Autonomous Database comes with predefined CPU/IO shares assigned to different consumer groups. You can set the consumer group to either low, medium or high while executing a data load job depending on your workload.

To link to files from a cloud store, do the following:

- Manage Cloud Storage Links for Data Link Jobs
- Prepare the Data Link Job
- Add Files or Folders for the Data Link Job
- Enter Details for the Data Link Job
- Run the Data Link Job
- View Details About the Data Link Job After It Is Run
- View the Table Resulting from the Data Link Job

- Manage Cloud Storage Links for Data Link Jobs
  Before you can link to data in a cloud store, you must establish a connection to the cloud store you want to use.
- Prepare the Data Link Job
- Add Files or Folders for the Data Link Job
- Enter Details for the Data Link Job
  Enter the details about the data link job in the Link Data from Cloud Storage pane.
- Run the Data Link Job
  Once you've added data sources to the data link cart and entered details about the data link job, you can run the job.
- View Details About the Data Link Job After It Is Run
  You can view the progress of the job on the Data Load dashboard.
- View the Table Resulting from the Data Link Job
  After running a data link job, you can view the table created by the data link job on the Data Load dashboard.

Manage Cloud Storage Links for Data Link Jobs

Before you can link to data in a cloud store, you must establish a connection to the cloud store you want to use.

On the Link Cloud Object page:

1. Click the Manage cloud store icon besides the field where you enter the cloud store location. Select **+ Create Cloud Store Location**.
2. Enter your information in the **Add Cloud Store Location** pane. See to add cloud storage location.

See Managing Connections.

To return to the Link Cloud Object page, click **Data Load** in the breadcrumbs at the top of the page and then navigate back to the page.

Prepare the Data Link Job

You may have to adjust your source data or your target table so that the source data links correctly to the external target table. Consider:

- If you're linking to multiple files, you must make sure that:
  - All the source files are of the same type, for example, CSV, JSON, etc.
  - The number, order, and data types of the columns in all the source files match.
- If you want to partition by date:
  - The source file must contain data where the data type is date or timestamp.
  - You must load a folder containing two or more data sources.
  - The names of the files in the folder must indicate a date or dates, for example, `MAR-1999.csv` or `2017-04-21.xlsx`.

Add Files or Folders for the Data Link Job

Add files from the cloud store to the data link cart, where you can edit the details of the data link job. To add the files:

1. From the list at the top of the navigator pane on the left, select the bucket with your source data.

   The list shows links that were established on the Manage Cloud Storage page. If you haven't yet registered the cloud store you want to use, click the **Connections** button under the Data Load menu in Data Studio suite of tools and register a connection.
2. Drag one or more items from the file navigator on the left and drop them into the cart on the right.
   - You can add files, folders, or both. A card is added to the cart for each file or folder you drag into it. The card lists the name of the source file or folder and a proposed name for the target table.
   - If you add a folder that contains multiple files, all the files must be of the same type, that is, CSV, TXT, etc.

     When you add the folder to the cart, a prompt is displayed that asks if you want to load all the objects from the multiple source files into a single target table. Click **Yes** to continue or **No** to cancel.
   - When you add multiple individual files or multiple folders to the cart, the data represented by each card will be loaded into a separate table, but all the items in the cart will be processed as part of the same data load job.
   - You can add files or folders from a different bucket, but if you do that, you're prompted to remove all files that are already in the cart before proceeding. To select files from a different bucket, select the bucket from the drop-down list in the navigator pane on the left and then add the file(s), as described above.
   - You can drop files or folders into the data load cart and then navigate away from the Data Link Object page. When you return to the page, those items remain on the page, but you may receive a message, "Remove All Data Link Items. Changing to another Cloud storage location requires all items to be removed from the data load job. Do you wish to continue?" Click **Yes** to remove the items from the cart. Click **No** to keep the items in the cart. Then you can continue to work.

You can remove items from the cart before running the data link job:

- To remove an item from the cart, select **Remove** on the card for the item Data Link cart menu bar at the top of the pane.

- To remove all items from the cart, click **Remove All** in the data link cart menu bar at the top of the pane.

Enter Details for the Data Link Job

Enter the details about the data link job in the Link Data from Cloud Storage pane.

On the card in the data link cart, click **Settings** to open the Link Data from Cloud Storage pane for that job. The pane contains:

- Settings Tab - Table Section
- Settings Tab - Properties Section
- Settings Tab - Mapping Section
- File Tab
- Table Tab
- SQL Tab
- Close Button - Save and Close the Pane

Settings Tab - Table Section

Set details about the target table in the **Table** section.

- **Name:** The name of the target table.
- **Partition Column:**

  List Partitions and Date-based partitions are the different types of partitions available in data linking.

  List partitioning is required when you specifically want to map rows to partitions based on discrete values.

  To partition according to a specific column, click the **Partition Column** drop-down list and select the column you want to use for the partitioning.

  You will have N files per partition value, all partitioned by the partition column you select.

  > **Note:**
  >
  > – For linked files (from external tables) there is also a requirement that for each file, the list partitioning column can contain only a single distinct value across all of the rows.
  >
  > – If a file is list partitioned, the partitioning key can only consist of a single column of the table.

  Date-based partitioning is available when you link a folder containing two or more data sources that have columns that contain date or timestamp data.

  To partition according to date, click the **Partition Column** drop-down list and select the **DATE** or **TIMESTAMP** column you want to use for the partitioning.

- **Validation Type:** Validation examines the source files, optional partitioning information, and report rows that do not match the format options specified. Select **None** for no validation; select **Sample** to perform validation based on a sample of the data; or select **Full** to perform validation based on all the data.

- **Use Wildcard:** This check box enables use of wildcard characters in search condition to retrieve specific group of files that matches the filter criteria.
  You can use a wildcard character, such as an asterisk (*) that searches, filters, and specifies groups of files that detect and add new files to the external table.

  For example, if you enter file*, then file01, file02, file03, and so on are considered to match the keyword. The asterisk (*) matches zero or more characters of the possibilities, to the keyword.

> **Note:**
>
> The wildcard support is incompatible with partitioning. The validation of source file fails if you use wildcards with partitioned data.

Settings Tab - Properties Section

Specify options to control how the source data is interpreted, previewed, and processed. These options vary, depending on the type of source data.

- **Encoding:** Select a character encoding type from the list. This option is available when the linked file is in plain text format (CSV, TSV, or TXT). The default encoding type is UTF-8.
- **Text enclosure:** Select the character for enclosing text: **"** (double-quote character), **'** (single-quote character) or **None**. This option is visible only when the selected file is in plain text format (CSV, TSV, or TXT).
- **Field delimiter:** Select the delimiter character used to separate columns in the source. For example, if the source file uses semicolons to delimit the columns, select **Semicolon** from this list. The default is **Comma**. This option is visible only when the selected file is in plain text format (CSV, TSV, or TXT).
- **Start processing data at row:** Specifies the number of rows to skip when linking the source data to the target external table:
  - If you select the **Column header row** option under **Source column name** (see below) and if you enter a number greater than 0 in the **Start processing data at row** field, then that number of rows after the first row are not linked to the target.
  - If you deselect the **Column header row** option under **Source column name**, and if you enter a number greater than 0 in the **Start processing data at row** field, then that number of rows including the first row are not linked to the target.
- **Source column name:** Select the **Column header row** checkbox to use the column names form the source table in the target table.
  - If you select this option, the first row in the file is processed as column names. The rows in the **Mapping** section, below, are filled with those names (and with the existing data types, unless you change them).
  - If you deselect this option, the first row is processed as data. To specify column names manually, enter a name for each target column in the **Mapping** section. (You will also have to enter data types.)
- **Numeric column**: Select the **Convert invalid data to null** checkbox to convert an invalid numeric column value into a null value.
- **Newlines included in data values**: Select this option if there are newline characters or returns to the beginning of the current line without advancing downward in the data fields. Selecting this option will increase the time taken to process the load. If you do not select

this option when loading the data, the rows with newlines in the fields will be rejected. You can view the rejected row in the Job Report panel.

Settings Tab - Mapping Section

The settings in the **Mapping** section control how data from the source files are linked to the rows of the target external table. For each row, the data from the column listed under **Source column** will be linked to the column listed under **Target column**.

- **Source column**: Lists the columns from the source file.

  If the **Column header row** option under **Properties** is selected, **Source column** shows the names of the columns in the source file. If the **Column header row** option is not selected, generic names like **COLUMN_1**, **COLUMN_2**, etc., are used. This field is always read only.

  You can view two source columns `FILE$NAME` and `SYSTIMESTAMP`. The `FILE$NAME` column enables you to locate the source file containing a particular data record. For example, you load a source file that contains a list of files. The file names in the file list refer to the department names across the organization. For instance, a *finance.txt* file contains data from the Finance department. In the mapping, you can use string data types to extract the department name from the output of the file name column. You can use the extracted department name to process data differently for each department.

  The `SYSTIMESTAMP` column allows us to view the current timestamp in the database.

  > **Note:**
  >
  > – `FILE$NAME` and `SYSTIMESTAMP` source columns are not included by default. You must check the **Include** check box and run the load for the target table to display these two columns.
  >
  > – When you are creating a livefeed, the `FILE$NAME` and `SYSTIMESTAMP` source columns appear in the Mapping table by default.

- **Target column:** Lists the columns in the target table.
  – If the **Column header row** option *is* selected, then the **Target column** uses the names of the columns in the source file. You can change the name of a target column by replacing the provided name with a new one. You need to make sure that the target column is not empty. Target column name must not be a duplicate of another target column. The target column name cannot have duplicate name as another target column. The target column length must not be beyond 128 bytes. 128 byte limit is a database limit.
  – If the **Column header row** option is *not* selected, then generic names like **COLUMN_1**, **COLUMN_2**, etc., are used. You can change the name of a target column by replacing the provided name with a new one.

  > **Note:**
  >
  > If you're linking multiple files from a folder in a single data link job, only the first file will be shown in the **Mapping** section. However, as long as the column names and data types match, the data from all source files will be linked.

- **Data Type:** Lists the data type to use for data in that column. The contents change depending on whether the **Get from file header** option is selected.
  - If the **Column header row** option *is* selected, **Data type** shows the data types of the columns in the source file. If you want to change the data type for the target, click the name and select a different one from the list.
  - If the **Column header row** option is *not* selected, **Data type** shows all available data types. Select the data type to use for the target column from the list.
- **Length/Precision** (Optional): For columns where the **Data Type** is **NUMBER**, enter the length/precision for the numbers in the column. Precision is the number of significant digits in a number. Precision can range from 1 to 38.

  For columns where Data Type is VARCHAR2, the **Auto** value in **Length/Precision** field enables the Auto Size feature.

  With the Auto-Size column Width feature, you can automatically size any column to fit the largest value in the column. Select **Auto** from the **Length/Precision** drop-down values or pick a value from the drop-down list.
- **Scale** (Optional): For columns where the **Data Type** is **NUMBER**, enter the scale for the numbers in the column. Scale is the number of digits to the right (positive) or left (negative) of the decimal point. Scale can range from ranges from -84 to 127.
- **Format:** If the data type in the **Data type** column is **DATE** or one of the **TIMESTAMP** types, select a format for that type from the from the **Format** drop-down list.

Preview Tab

The **Load Preview** menu in the **Preview** tab displays the source data in tabular form. The display reflects the settings you chose in the **Properties** section. The **File** menu displays source data with the column names.

If you dragged a folder containing multiple files into the data link cart and then clicked  **Settings** for that card, the **Preview** pane includes a **Preview Object (File)** drop-down list at the top of the pane that lists all the files in the folder. Select the source file you want to preview from that list.

Table Tab

The **Table** tab displays what the target table is expected to look like after the data has been linked.

SQL Tab

The **SQL** tab displays the SQL commands that will be run to complete this data link job.

> **Note:**
>
> You can see the SQL code even before the table is created.

Close Button - Save and Close the Pane

After entering all the details for the data link job, click **Close** at the bottom of the page. This saves the details you entered and returns you to the Link Data from Cloud Storage pane.

Run the Data Link Job

Once you've added data sources to the data link cart and entered details about the data link job, you can run the job.

To run the job:

1. If you haven't already done so, click the **Close** button in the **Link Data from Cloud Storage** pane to save your settings and close the pane. If any of the settings are invalid, an error message reports the problem. Fix the problem and click **Close**.

2. Click  **Start** in the data link cart menu bar. To stop the data link job, click  **Stop**.

   When the data link job completes, the **Data Load Dashboard** page displays the results of the job under **Table and View Loads** section.

   Once the data link job starts, you can view the progress of the job in the Data Load dashboard.

View Details About the Data Link Job After It Is Run

You can view the progress of the job on the Data Load dashboard.

When the data load job completes, the Data Load dashboard page displays the results of the job. At the top of the header of the table load, you can view the name of the table along with the total columns present in the table.

Click **Job Report** to view the total number of rows processed successfully and the count of rejected rows. You can also view the Start time. The SQL pane of the Job Report displays the equivalent SQL code of the job.

To view information about an item in the job, click the **Actions** icon on the Table Load.

To view a log of the load operation, click the Logging icon. You can save the log, clear it, or refresh it. Click **OK** to dismiss the log.

View the Table Resulting from the Data Link Job

After running a data link job, you can view the table created by the data link job on the Data Load dashboard.

Fix your data load job. After your data load job, you might see errors that you want to correct, or upon inspection, realize that you wanted to name a column differently. In such cases, click the **Reload** option on the selected Table Load to reload cards from your recent cart and edit them as you did before your first attempt. The Reload icon reloads the source data with the fixes suggested by the tool. Click the **Actions** icon on the Table header, click **Table** and select **Edit** to make any changes to the data load job (i.e., change a column name).

Feeding Data

You can run a live table feed on demand, on a schedule, or as the result of a notification.

A Live Table Feed automates loading of data into a table in your database. Files automatically load as they appear in object storage and the Live Table Feed system ensures that files are only loaded once. The loading can happen manually, by a schedule, or even by notifications delivered directly from Object Storage.

The bucket can contain files in these formats: AVRO, CSV, JSON, GeoJSON, Parquet, ORC, Delimited TXT. All of the files must have the same column signature.

About the Live Feed Page

On the Database Actions - Data Load Dashboard page select **FEED DATA** to display the Live Feed page. On this page, you can:

- Manage Cloud Storage Connections for Live Table Feeds
- Create a Live Table Feed Object
- List, Filter, and Sort Live Table Feed Objects
- Find and View Live Table Feed Objects
- Run a Live Table Feed
- Delete a Live Table Feed

Manage Cloud Storage Connections for Live Table Feeds

Before you create a live table feed, you must establish a connection to the cloud store you want to use:

1. Click **Connections** under the Data Load menu. For instructions, see Managing Connections.

Create a Live Table Feed Object

To create a live table feed object,

1. On the Live Feed page, click the **+ Create Live Table Feed** button to display the Live Feed Settings pane. Enter information on the Data Source tab as follows:
   - **Cloud Store Location**: Select the Cloud Store Location from the drop-down. Select the cloud connection for the bucket containing the file you want to use for feeding data.
   - **Object Filter (glob):** Enter a file glob to limit the live table feed to only those files in the bucket that match the glob. For example, to limit the files to only those that are CSV files with names that start with SALES, enter SALES*.CSV.
   - Under the Live Feed File Preview section, specify:
   - **File Preview**: It displays a preview of the file you select in the previous step.

   Click **Next** to progress to the Table Settings tab.

   Target Table Name: Accept the default name or enter a different name. This is the name of the target table that data from the live feed will be loaded into in your Autonomous Database instance. If the table does not exist, Live Feed will attempt to guess the correct columns. You can pre-create the table in which you wish the Live Feed to load into. This is for higher accuracy.

   The Table Settings tab specifies options to control how the source data is interpreted, previewed, and processed. These options vary, depending on the type of source data.
   - **Encoding:** Select a character encoding type from the list. This option is available when the linked file is in plain text format (CSV, TSV, or TXT). The default encoding type is UTF-8.
   - **Text enclosure:** Select the character for enclosing text: **"** (double-quote character),**'** (single-quote ch **None**. This option is visible only when the selected file is in plain text format (CSV, TSV, or TXT).
   - **Field delimiter:** Select the delimiter character used to separate columns in the source. For example, if the source file uses semicolons to delimit the columns, select **Semicolon** from this list. The default is **Comma**. This option is visible only when the selected file is in plain text format (CSV, TSV, or TXT).
   - **Start processing data at row:** Specifies the number of rows to skip when linking the source data to the target external table:

If you select the **Column header row** option under **Source column name** (see below) and if you enter a number greater than 0 in the **Start processing data at row**field, then that number of rows after the first row are not linked to the target.

If you deselect the **Column header row** option under **Source column name**, and if you enter a number greater than 0 in the **Start processing data at row** field, then that number of rows including the first row are not linked to the target.

Column header row: Select the **Column header row** checkbox to use the column names form the source table in the target table.

If you select this option, the first row in the file is processed as column names. The rows in the **Mapping** section, below, are filled with those names (and with the existing data types, unless you change them).

If you deselect this option, the first row is processed as data. To specify column names manually, enter a name for each target column in the **Mapping** section. (You will also have to enter data types).

Select the **Convert invalid data to null** checkbox to convert an invalid numeric column value into a null value.

- Edit or update the table settings in the Table Settings section: In this pane, the mapping of the source to target columns is displayed.
  - Select the Include check box at the beginning of a row to add the column to the target table.
  - Select or enter values for column attributes such as Target Column Name, Column Type, Precision, Scale, Default, Primary Key and Nullable.
  - You need to review the suggested data type and if needed, modify it by entering the data type directly into the target cell.
    Review the generated mapping table code based on the selections made in the previous screens.

  Click **Next** to progress to the Preview tab.
- The Preview Pane displays the changes you make to the table.
- Click **Next** to progress to the Live Feed Settings tab.

On the Live Settings tab, specify the following field values:

- **Live Table Feed Name**: Accept the default name or enter a different name to identify this live table feed.
- **Enable for Notification**: Select this option so that new or changed data in the data source will be loaded based on an Oracle Cloud Infrastructure *notification*. When you select this option, you can avoid delays that might occur when polling is initiated on a schedule (that is, if you selected the live table feed **Scheduled** option).

  When you select the **Enable for Notification** option, you must also:
  - Configure your object store bucket to emit notifications
  - Create a Notifications service subscription topic
  - Create an Events service rule
  - Copy the notification URL
  - Create a Notifications service subscription
  - Confirm that notifications are allowed

  For complete instructions, see Creating a Notification-Based Live Table Feed.

- **Enable for Scheduling:** Select this option to set up a schedule for running the live table feed object; that is, to poll the data source on a regular basis:
  - In the time interval fields, enter a number, and select a time type and the days on which to poll the bucket for new or changed files. For example, to poll every two hours on Monday, Wednesday, and Friday, enter **2**, select **Hours**. You can select **All Days**, **Monday to Friday**, **Sunday to Thursday**, or **Custom** from **Week Days** drop-down. The Custom field enables you to select **Monday**, **Tuesday**, **Wednesday**, **Thursday** and **Friday** in the appropriate fields.
  - Select a start and end date with start and end time. If you don't select a start date, the current time and date are used as the start date. The end date is optional. However, without an end date, the live feed will continue to poll.

  Select a consumer group from the drop-down, namely, low, medium and high.

2. Click **Create** to create the Live Table feed object.

List, Filter, and Sort Live Table Feed Objects

When you open the Live Feed page, existing live table feed objects are displayed as cards on the page. They are identified as LIVE_TABLE_FEED entities.

To filter live table feed objects:

1. Click the search field at the top of the page to display filter options. By default, the live table feed objects from the current user's schema are shown. As soon as you start typing in the search field, the feed tool returns the values which contain the letters you type. You can remove the filter by deleting the content from search box and clicking the cross icon that appears next to the search box.
2. To include objects from other schemas, select the drop-down next to the search field, under **Schema**. To remove a schema from the filter list, deselect the box next to its name.
3. To show objects from all available schemas, select **All** from the **Schema** drop-down.

To sort live table feed objects

1. Click the **Sort by** button at the top right of the page.
2. Select a sorting option. To sort ascending, click the icon with the up arrow. To sort descending, click the icon with the down arrow.

Find and View Live Table Feed Objects

To search for available live table feed entities in the selected schemas, enter a value in the search field at the top of the page and press **Enter**. The display then includes only the entities whose names contain the characters in the search field. To clear the search field, click the Clear search results (X) icon in the search field.

To remove a schema or sorting value from the selected filters, deselect the schema or sorting value in the filter panel, or click the Remove filter (X) icon for the schema or sorting value above the display of live table feed objects. To close the filter panel, click the Hide filter panel (X) icon in the panel.

To refresh the display of live table feeds, click the Refresh icon at the top of the page.

Edit a Live Table Feed Object

To edit details of a live table feed object,

1. On the Live Feed page, find the card for the live table feed whose details you want to edit.

2. Click the Actions icon (three dots) on the card and select **Edit Live Table Feed**. You can edit the following options:

   - **Object Filter (Regular Expression):** Enter a regular expression to limit the live table feed to only those files in the bucket that match the expression. For example, to limit the files to only those that are CSV files with names that start with SALES, enter `SALES.\*.CSV`.
   - **Enable for Notification**: Select this option so that new or changed data in the data source will be loaded based on an Oracle Cloud Infrastructure *notification*. When you select this option, you can prevent any delays that might occur when polling is initiated on a schedule (that is, the live table feed **Scheduled** option).

     When you select the **Enable for Notification** option, you must also:

     – Copy the live table feeds notification URL
     – Configure your cloud store to emit notifications
     – Configure Oracle Cloud Infrastructure to route events to the endpoint used for the live table feed.
     – Create a rule.
     – Create a subscription.
     – Confirm that notifications are allowed at the live feed service.

     For complete instructions, see Creating a Notification-Based Live Table Feed.
   - **Scheduled:** Select this option to set up a schedule for running the live table feed object; that is, to poll the data source on a regular bases:

     – In the time interval fields, enter a number, and select a time type and the days on which to poll the bucket for new or changed files. For example, to poll every two hours on Monday, Wednesday, and Friday, enter `2`, select **Hours**. You can select **All Days**, **Monday to Friday**, **Sunday to Thursday**, or **Custom** from **Week Days** drop-down. The Custom field enables you to select **Monday**, **Tuesday**, **Wednesday**, **Thursday** and **Friday** in the appropriate fields.
     – Select a start and end date with start and end time.

3. Click **Save**.

Run a Live Table Feed

You can run a live table feed on demand, on a schedule, or as the result of a notification.

To run a live table feed on demand:

1. On the Live Feed page, find the card for the live table feed you want to run.
2. Click the Actions icon (three dots) on the card and select **Run Live Table Feed Immediately (Once)**.

To run a live table feed on a schedule:

You can set a schedule for running live table feeds on the **Create Live Table Feed** pane (when creating a new table feed) or the **Edit Live Table Feed** pane (when editing an existing table feed). See Create a Live Table Feed Object or Edit a Live Table Feed Object.

To run a live table feed as the result of a notification:

See Creating a Notification-Based Live Table Feed.

Select the **Scheduled** check box to display the schedule options and then set the schedule by selecting the options you want.

To view live table feed run details:

1. On the Live Feed page, find the card for the live table feed whose run details you want to see.
2. Click the Actions icon (three dots) on the card and select **Live Table Feed Run Details**. The **Objects** tab on the Live Table Feed Run Details pane displays information about the jobs, such as when the run occurred, the objects involved in the run, the table owner, the table Name, status of the live feed, the rows loaded and the rows rejected, and other details. Click the **All** tab to view more details, such as the event type.

Delete a Live Table Feed

1. On the Live Feed page, find the card for the live table feed job you want to delete.
2. Click the Actions icon (three dots) on the card and select **Delete Live Table Feed**.

- Creating a Notification-Based Live Table Feed
  You can load data through a live table feed based on an Oracle Cloud Infrastructure *notification*.
- Creating a Notification-Based Live Table Feed using Amazon Simple Storage Service (S3)
  You can integrate Amazon Simple Storage Service (S3) and Oracle Cloud Infrastructure (OCI) to automate the process of live feed notifications when storage objects it is observing have updates. The following section provides instructions for creating event notifications in your Amazon S3 bucket where your data files are stored.
- Creating a Notification-Based Live Table Feed using Microsoft Azure
  A notification-based Live Table Feed is an interface between Oracle Cloud Infrastructure and a third-party cloud message queuing service such as Azure Event Grid.

Creating a Notification-Based Live Table Feed

You can load data through a live table feed based on an Oracle Cloud Infrastructure *notification*.

In addition to being able to run a live table feed on demand or on a schedule, as described in Feeding Data, you can also run a feed as the result of a notification. When data in the source bucket is changed, a notification is sent which triggers a run of the table feed. With a notification-based live table feed, you can avoid any delay that might come from running on-demand or scheduled live table feed jobs.

> **Note:**
>
> - Notification-based live table feeds aren't available on the Oracle Cloud Infrastructure free tier. You must be on a paid tenancy with appropriate permissions on your account to use this feature.
> - Notification-based live table feeds aren't available on Oracle Autonomous Data Warehouse Databases (ADW) that are configured using a private endpoint.

To create a notification-based live table feed:

- Step 1: Configure your object store bucket to emit notifications

- Step 2: Create a Notifications service subscription topic
- Step 3: Create an Events service rule
- Step 4: Create and configure a live table feed to use notifications, and copy the notification URL
- Step 5: Create a Notifications service subscription
- Step 6: Confirm that the endpoint can receive notifications

> **Tip:**
>
> To complete those steps, you will alternate between Oracle Cloud Infrastructure Console pages and Oracle Database Actions pages. You may find it convenient to open the Cloud Console in one browser page or tab and Database Actions in another, so it's easy to move back and forth.

Step 1: Configure your object store bucket to emit notifications

Where: Oracle Cloud Infrastructure Console: Object Storage & Archive Storage - Buckets page

Configure the bucket containing your source data so that it will emit notifications when the data changes. You can set this option when you create a bucket or you can set it in an existing bucket.

1. Open the Cloud Console navigation menu and click **Storage**. Under **Object Storage and Archive Storage**, click **Buckets**.
2. **If you're creating a new bucket:**
   a. On the Buckets page, click the **Create Bucket** button to create a new bucket, as described in Managing Buckets. In the **Create Bucket** wizard, select the **Emit Object Events** option, along with the other options for your new bucket.
   b. Click **Create**.

   If you're using an existing bucket:
   a. On the Buckets page, click the name of the bucket you want to use, as described in Managing Buckets.
   b. On the Bucket Details page, click the **Edit** link next to **Emit Object Events**.
   c. Select the **Emit Objects Events** check box, and then click **Save Changes**.

Step 2: Create a Notifications service subscription topic

Where: Oracle Cloud Infrastructure Console: Notifications - Topics page

1. Open the Cloud Console navigation menu and click **Developer Services**. Under **Application Integration**, click **Notifications**.
2. Click **Create Topic**, enter a name and optional description, and then click **Create**.

Step 3: Create an Events service rule

Where: Oracle Cloud Infrastructure Console: Events - Rules page

1. Open the Cloud Console navigation menu and click **Observability & Management**. Under **Events Service**, click **Rules**.

2. Click **Create Rule**, and fill out the Create Rule page as described in Managing Rules for Events.
   - Under **Rule Conditions**, select:
     - **Condition:** Event Type
     - **Service Name:** Object Storage
     - **Event Type:** Object - Create
   - Under **Actions**, select:
     - **Action Type:** Notifications
     - **Notifications Compartment:** Select the compartment to use for the notifications.
     - **Topic:** Select the name of the topic you created above, in Step 2: Create a Notifications service subscription topic.
3. Click **Create Rule**.

Step 4: Create and configure a live table feed to use notifications, and copy the notification URL

Where: Database Actions: Live Feeds page

You can configure a new or an existing live table feed to use notifications:

1. Go to the Database Actions Live Feeds page, as described in Feeding Data.
2. Create or edit a live table feed object, as described in Create a Live Table Feed Object or Edit a Live Table Feed Object. Select the **Enable for Notification** option
3. Click **Create** or **Save**.
4. Click the Actions (three vertical dots) icon on the card for your live feed, and select **Show Confirmation URL**.
5. In the **Notification URL** dialog box, click the **Copy** icon to copy the URL to the clipboard. You may want to copy it to a temporary file, so you can retrieve it later. You'll use this URL in the next step, Step 5: Create a Notifications service subscription.

Step 5: Create a Notifications service subscription

Where: Oracle Cloud Infrastructure Console: Notifications - Subscriptions page

1. Return to the Oracle Cloud Infrastructure Console. Open the navigation menu and click **Developer Services**. Under **Application Integration**, click **Notifications**.
2. On the Notifications page, click the **Subscriptions** tab (on the left side of the page), the status will be **Active**.
3. Click **Create Subscription** and fill in the **Create Subscription** page:
   - **Subscription topic:** Select the subscription topic you created in Step 2: Create a Notifications service subscription topic.
   - **Protocol:** HTTPS (Custom URL)
   - **URL:** Paste in the URL you copied in Step 4: Create and configure a live table feed to use notifications, and copy the notification URL.
   - Click **Create**. The subscription will be listed in the **Subscriptions** table in a state of "Pending."

Step 6: Confirm that the endpoint can receive notifications

Where: Database Actions: Live Feeds page

1. Return to the Database Actions Live Feeds page and find the card for the live table feed you are configuring for a notification-based feed.
2. Click the Actions (three vertical dots) icon on the card, and select **Show Confirmation URL**.
3. In the **Confirmation URL** dialog box, click the link to confirm the URL. This does not close this dialog box. If the link is successful, a message is displayed that confirms the subscription is active.
4. Return to the **Confirmation URL** dialog box and select the **Check only when the cloud store confirmation process is complete** check box, and click **OK**.

Once you finish the above steps, any new files uploaded to the bucket will automatically be loaded into the live table feed table.

Creating a Notification-Based Live Table Feed using Amazon Simple Storage Service (S3)

You can integrate Amazon Simple Storage Service (S3) and Oracle Cloud Infrastructure (OCI) to automate the process of live feed notifications when storage objects it is observing have updates. The following section provides instructions for creating event notifications in your Amazon S3 bucket where your data files are stored.

> **Tip:**
>
> To complete these steps, you will need to alternate between Amazon Web Services (AWS) Management console and Oracle Database Actions pages. You may find it convenient to open the Amazon Web Services in one browser page or tab and Database Actions in another, so it is easy to move back and forth.

To create a notification- based live feed with Amazon S3 as cloud storage you must:

- Step 1: Create your object store bucket in Amazon S3
- Step 2: Create Access Keys
- Step 3: Add an OCI Cloud Storage using Amazon S3
- Step 4: Create and configure a live table feed to use notifications and copy the notification URL
- Step 5: Create a notifications service subscription topic
- Step 6: Enable and configure event notifications using the Amazon S3 console
- Step 7: Create a notifications service subscription
- Step 8: Confirm that the endpoint can receive notifications

Step 1: Create your object store bucket in Amazon S3

Where: Amazon Web Services (AWS) Management console

Configure and create your bucket containing source data so that it emits notifications when the data changes.

1. Log in to AWS Management console and open the Amazon S3 console.

2. On the home page click the **Create Bucket** icon.
3. In **Bucket name**, enter a valid name for your bucket. For example: testbucket. After you create the bucket, you cannot change its name.
4. In **Region**, select the Amazon Web Services (AWS) Region from the dropdown. For example: us-west-2
5. In Bucket settings for Block Public Access, select the **Block Public Access** settings that you want to apply to the bucket. It is recommended to keep all settings enabled unless you know that you need to turn any of them off.
6. Select **Advanced settings**, and accept all the default options if you want to enable S3 Object Lock. This step is optional.
7. Select **Create bucket**.

Step 2: Create Access Keys

Where: AWS Management console

To access Amazon Simple Notification Service (SNS), you must have credentials that Amazon Web Services (AWS) can use to validate your requests. These credentials must have permissions to access Amazon SNS topics. The following steps provide you details on steps to create access keys using AWS Identity and Access Management (IAM) for security purposes.

1. Log in to AWS Management console and open Amazon Identity and Access Management (IAM) console.
2. On the navigation menu, select **Users**.
3. Select your **user name**.
4. In the Security Credentials tab, select **Create access key**.
5. Copy the **Access key ID** and **Secret access key** in the display. Paste them in a clipboard.
6. To download the keys, select **Download.csv file** icon. This way you can store the file in a secure location.

Step 3: Add an Amazon S3 Cloud Storage Link

Where: Database Actions: Manage Cloud page

Before you create a live table feed, you must establish a connection to the cloud store you want to use.

1. Click the **Manage Cloud Store** button at the top of the page to go to the Manage Cloud page. For further instructions on adding source files residing in cloud storage provided by Amazon S3, refer to *Create an Amazon S3 Cloud Storage Link* topic in Managing Connections.

> **Note:**
>
> Paste the Access key ID and Secret access key generated in the previous step (Step 2: Create Access Keys) to their respective text fields in the **Add Cloud Storage** page.

Step 4: Create and configure a live table feed to use notifications, and copy the notification URL

Where: Database Actions: Live Feeds page

Creating a live table feed enables you to load data in real time from external storage sources to your table in ADB. External storage you use include as Oracle Object Store, AWS S3 or Microsoft Azure containers.

You can configure a new or an existing live table feed to use notifications:

1. Go to the Database Actions Live Feeds page, as described in Feeding Data.
2. Create or edit a live table feed object, as described in Create a Live Table Feed Object or Edit a Live Table Feed Object. Select the **Enable for Notification** option
3. Click **Create** or **Save**.
4. Click the **Actions** (three vertical dots) icon on the card for your live feed, and select **Show Notification URL**.
5. In the **Notification URL** dialog box, click the **Copy** icon to copy the URL to the clipboard. You may want to copy it to a temporary file, so you can retrieve it later. You will use this URL in the subsequent step (Step 7: Create a notifications service subscription).

Step 5: Create a notifications service subscription topic

Where: Amazon Simple Notification Service (SNS) console

You receive Amazon S3 notifications using Amazon Simple Notification Service (Amazon SNS) topic. You need to add a notification configuration to your bucket using an Amazon SNS topic. SNS topics are shared locations which are used to send notifications of various events that happen in AWS buckets.

During creation, you select a topic name and topic type. After creating a topic, you cannot change the topic type or name. All other configuration choices are optional during topic creation, which you can edit later.

To access any AWS service, you must first create an AWS account.

Navigate to the AWS Management console, and then select **Create an AWS Account**.

Follow the instructions as provided in the Amazon SNS link to create your first IAM administrator user and group. Now you can log in to any of the AWS services as an IAM user.

1. Log in to Amazon SNS console as an IAM user.
2. On the **Topics** page, select **Create topic**.
3. Specify the following fields on the **Create topic** page, in the **Details** section.
   - **Type:**Standard (Standard or FIFO)
   - **Name:** notify-topic. For a FIFO topic, add fifo to the end of the name.
   - **Display Name:** This field is optional.
4. Expand the **Encryption** section and select **Disable encryption**.
5. Expand the **Access policy** section and configure additional access permissions, if required. By default, only the topic owner can publish or subscribe to the topic. This step is optional. Edit the JSON format of the policy based on the topic details you enter. Here is a sample of Access policy in JSON format.

```
{ "Version": "2008-10-17",
"Id": "__default_policy_ID",
"Statement":[
{"Sid": "__default_statement_ID",
"Effect": "Allow",
```

```
"Principal": {"AWS": "*"
},"Action": [
        "SNS:Publish",
        "SNS:RemovePermission",
        "SNS:SetTopicAttributes",
        "SNS:DeleteTopic",
        "SNS:ListSubscriptionsByTopic",
        "SNS:GetTopicAttributes",
        "SNS:AddPermission",
        "SNS:Subscribe"
      ],
      "Resource": "arn:aws:sns:us-west-2:555555555555:notify-topic", //us-
west-2 is the region
      "Condition": {
        "StringEquals": {
          "AWS:SourceOwner": "555555555555"
        }
      }
    },
    {
      "Sid": "s3_policy", //This field accepts string values
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": [
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:us-west-2:555555555555:notify-topic", //
notify-topic is the topic name
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "555555555555" //This is the Account ID
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:*:*:testbucket /*testbucket is the
s3 bucket name. You will get notifications only when file is uploaded to
this
    bucket.*/
"
        }
      }
    }
  ]
}
```

6. Expand the **Delivery retry policy (HTTP/S)** section to configure how Amazon SNS retries failed message delivery attempts. This step is optional.
7. Expand the **Delivery status logging** section to configure how Amazon SNS logs the delivery of messages to CloudWatch. This step is optional.
8. Expand **Tags** section to add metadata tags to the topic. This step is optional.
9. Select **Create topic**.
10. The topic's Name, ARN (Amazon Resource Name), and Topic owner's AWS account ID are displayed in the Details section.

11. Copy the topic ARN to the clipboard.

Step 6: Enable and configure event notifications using the Amazon S3 console

Where: Amazon S3 Management console

You can enable Amazon S3 bucket events to send a notification message to a destination whenever those events occur. You configure event notifications for your S3 bucket to notify OCI when there is an update or new data available to load. The following steps explain the procedure to be followed in Amazon S3 console to enable event notifications.

1. Log in to Amazon S3 Management console and sign in as an IAM (Amazon Identity and Access Management) user.
2. In the **Buckets** list, select the name of the bucket i.e. testbucket. This is the bucket that you had created in Step 1: Create your object store bucket in Amazon S3.
3. Select **Properties** icon.
4. Navigate to the **Event Notifications** section and select **Create event notification** icon.
5. In the **General configuration** section, specify the following values for event notification.
   - **Event name:** bucket-notification
   - **Prefix:** This value is to filter event notifications by prefix. It is an optional value. This is added to filter event activity.
   - **Suffix:** This value is to filter event notifications by suffix. It is an optional value. This is added to filter event activity.
6. In the **Event types** section, select one or more event types that you want to receive notifications for. If you are unsure of what event types to pick, then select the **All object create events** option.
7. In the **Destination** section, select **SNS Topic** as the event notification destination.

> **Note:**
>
> Before you can publish event notifications, you must grant the Amazon S3 the necessary permissions to call the relevant API. This is so that it can publish notifications to a Lambda function or an SNS topic.

8. After you select SNS topic as the event notification destination, select the SNS topic i.e. notify-topic from dropdown. This is the topic you created in Step 5: Create a notifications service subscription topic.
9. Select **Save changes**.

Step 7: Create a notifications service subscription

Where: Amazon SNS console

Every Amazon SNS topic has a set of subscriptions. Once a message is published to a topic, SNS handles distributing the message to all its subscribers. The subscribers can be AWS Lambda functions, HTTP(S) endpoints, email addresses and mobile phone numbers capable of receiving SMS messages.

Amazon SNS matches the topic to a list of subscribers who have subscribed to that topic and delivers the message to each of those subscribers.

1. Log in to Amazon SNS console.

2. In the left navigation pane, select **Subscriptions**.
3. Select **Create subscription** on the subscriptions page.
4. In the **Details** section of the **Create subscription** page, specify the following values.
   - **Topic ARN:** Paste the ARN value copied from previous step (Step 5: Create a notifications service subscription topic).
   - **Protocol:** HTTPS
   - **Endpoint:** Paste the endpoint value you copied while creating the live table feed in previous step (Step 4: Create and configure a live table feed to use notifications and copy the notification URL).
5. Expand the **Subscription filter policy** section to configure a filter policy. This step is optional.
6. Expand the **Redrive policy (dead-letter queue)** section to configure a dead-letter queue for the subscription. This step is optional.
7. Select **Create subscription**.

> **Note:**
>
> HTTP(S) endpoints, email addresses, and AWS resources in other AWS accounts require confirmation of the subscription before they can receive messages.

Step 8: Confirm that the endpoint can receive notifications

Where: Database Actions: Live Feeds page

1. Return to the Database Actions Live Feeds page and find the card for the live table feed you are configuring for a notification-based feed.
2. Click the **Actions** (three vertical dots) icon on the card, and select **Show Confirmation URL**.
3. In the **Confirmation URL** dialog box, click the link to confirm the URL. This does not close this dialog box. If the link is successful, a message is displayed that confirms the subscription is active.
4. Return to the **Confirmation URL** dialog box and select the Check only when the cloud store confirmation process is complete check box, and click **OK**.

Once you finish the above steps, any new files uploaded to the bucket will automatically be loaded into the live table feed table.

For more information on how to enable and configure event notifications using the Amazon S3 console, refer Enabling and configuring event notifications using the Amazon S3 console.

Creating a Notification-Based Live Table Feed using Microsoft Azure

A notification-based Live Table Feed is an interface between Oracle Cloud Infrastructure and a third-party cloud message queuing service such as Azure Event Grid.

The following section explains the procedure to generate automatic live feed messages using Microsoft (MS) Azure as the cloud storage. When there is an update in the container and the notification conditions are met, a log message is generated and displayed in the live feed in Oracle Cloud Infrastructure.

To create a notification-based live feed with Microsoft Azure as cloud storage you must:

- Step 1: Create a resource group in Microsoft Azure
- Step 2: Create a storage account in Microsoft Azure
- Step 3: Create Access Keys
- Step 4: Create a container
- Step 5: Add cloud storage using Microsoft Azure cloud store
- Step 6: Create and configure a live table feed to use notifications and copy the notification URL
- Step 7: Enable Event Resource Provider
- Step 8: Create Event subscription
- Step 9: Confirm that the endpoint can receive notifications

> **Tip:**
>
> To complete the steps above, you will need to alternate between Microsoft Azure portal and Oracle Database Actions pages. You may find it convenient to open the Microsoft Azure portal in one browser page or tab and Database Actions in another, so it is easy to move back and forth.

Step 1: Create a resource group in Microsoft Azure

Where: Microsoft Azure Portal

Resource groups are logical containers where you can manage Azure resources like storage accounts. Resource groups are created so you can deploy, update and delete them as a group. You can create a resource group by following these steps:

1. On the Azure portal, click the **Resource groups** button.
2. Select **Add**.
3. Enter the following values:
   - **Subscription:** Select your Azure subscription, such as Microsoft Azure Enterprise.
   - **Resource group:** Enter a new resource group name, such as resource-group.
   - **Region:** Select your location, such as US west.
4. Click **Review+create**.
5. Click **Create**. It takes a few seconds to create a resource group.

Step 2: Create a storage account in Microsoft Azure

Where: Microsoft Azure Portal

An Azure storage account contains all your storage data objects like blobs, tables, disks etc. You can create a storage account inside the resource group. It provides a unique namespace for your data. To create a storage account, do the following:

1. From the left portal menu, select **Storage accounts** to display a list of your storage accounts.
2. On the Storage accounts page, click the **Create** icon.

3. On the Basic tab, provide the following information for your storage account.
   - **Subscription:** Microsoft Azure Enterprise
   - **Resource group:** resource-group
   - **Storage account name:** teststorage
   - **Region:** Select your location, such as US west.
   - **Redundancy:** Locally-redundant storage (LRS)
4. You can select **Review+create** to accept the default options and proceed to validate the account.
5. After the validation passes, you can proceed to click on **Create storage** account. In case the validation fails, the portal indicates which settings must be modified.

Step 3: Create access Keys

Where: Microsoft Azure Portal

You must grant Microsoft Azure the permissions necessary to obtain access keys on your storage locations. The access keys specific to the storage account are generated automatically after the storage account is created in the previous step. The following steps describes the procedure to create access keys.

1. In **Security+Networking** , select **Access keys**. Your account access keys appear with the complete connection string for each key.
2. Select **Show keys** to show your access keys and connection string for each key and to copy values.
3. Copy the connection string value under key1. This value will be pasted in **Azure storage account access key** text field of next step (Step 5: Add cloud storage using Microsoft Azure cloud store).
4. Copy the storage account name i.e. teststorage and paste it in **Azure storage account name** text field of next step (Step 5: Add cloud storage using Microsoft Azure cloud store).
5. Test the credentials to see if it works or not.

Step 4: Create a container

Where: Microsoft Azure Portal

A container is a location (also known as buckets in Amazon S3 and OCI) which holds Azure Blob (Binary large object) storage. Follow these steps to create a container.

1. Navigate to your new storage account in the Azure portal.
2. In the left menu for the storage account, scroll to the **Data storage** section, then select **Containers**.
3. Click the **+Container** icon.
4. Enter the name for your new container. The container name must be lowercase, must start with a letter or number, and can include only letters, numbers, and the dash character.
5. Set the level of **Public Access Level** to Private. The default level is Private.
6. Select **Create** to create the container.

Step 5: Add cloud storage using Microsoft Azure cloud store

Where: Database Actions: Manage Cloud page

1. Click the **Manage Cloud Store** button at the top of the page to go to the Manage Cloud page. For further instructions on adding source files residing in cloud storage provided by Microsoft Azure cloud storage, refer to *Create an Microsoft Azure Cloud Storage Link* topic in Managing Connections section.

> **Note:**
>
> Paste the connection string value under key 1 of previous step Step 3: Create Access Keys in **Azure storage account access key** text field of **Add Cloud Storage** page. Also paste the storage account name generated in the previous step Step 3: Create Access Keys in **Azure storage account name** text field of **Add Cloud Storage** page.

Step 6: Create and configure a live table feed to use notifications and copy the notification URL

Where: Database Actions: Live Feeds page

The Live table feed object enables data to be loaded from Microsoft Azure cloud storage with no polling delay. This object creates an integration between Oracle Cloud Interface and Microsoft Azure.

You can configure a new or an existing live table feed to use notifications:

1. Go to the Database Actions Live Feeds page, as described in Feeding Data.
2. Create or edit a live table feed object, as described in Create a Live Table Feed Object or Edit a Live Table Feed Object. Select the **Enable for Notification** option
3. Click **Create** or **Save**.
4. Click the **Actions** (three vertical dots) icon on the card for your live feed, and select **Show Confirmation URL**.
5. In the **Notification URL** dialog box, click the **Copy** icon to copy the URL to the clipboard. You may want to copy it to a temporary file, so you can retrieve it later. You will use this URL in the subsequent step, (Step 8: Create Event subscription).

Step 7: Enable Event Resource Provider

Where: Microsoft Azure Portal

If this is the first time you are using Event Grid, you must enable Event Grid resource provider.

1. Select **Subscriptions** on the left menu.
2. Select the subscription you are using for Event Grid i.e. Microsoft Azure Enterprise.
3. On the left menu, under Settings, select **Resource Providers**.
4. Search **Microsoft.EventGrid**.
5. Select **Register**.

It takes a minute for the registration to finish.

Step 8: Create Event subscription

Where: Microsoft Azure Portal

You create an Event Subscription by configuring the subscription and specifying the endpoint that will receive the notifications.

1. Select the storage account you created in Step 2: Create a storage account in Microsoft Azure.
2. Select the **Events** icon in the left navigation pane.
3. Click on **+Event Subscription**.

The **Create Event Subscription** window appears.

1. Specify the following fields in **Event Subscription** details section:
   - **Name:** Eventssub. This is the name of the Event Subscription we create.
   - **Event Schema:** Event Grid Schema
2. Specify the following fields in **Topic Details** section:
   - **Topic Type:** Storage account
   - **System Topic Name:** eventtopic.
3. Specify the following fields in **Event Types** section:
   - **Event Type:** MicrosoftStorage.BlobCreated
4. Specify the following fields in **Endpoint Details** section:
   - **Endpoint Type:** Web Hook
   - **Endpoint:** Paste the Notification URL copied in Step 6: Create and configure a live table feed to use notifications and copy the notification URL.
5. Select **Create**.

This way Microsoft Azure creates a system topic first and then the Event Subscription for the topic.

Step 9: Confirm that the endpoint can receive notifications

Where: Database Actions: Live Feeds page

1. Return to the Database Actions Live Feeds page and find the card for the live table feed you are configuring for a notification-based feed as created in Step 6: Create and configure a live table feed to use notifications and copy the notification URL.
2. Click the **Actions** (three vertical dots) icon on the card, and select **Show Confirmation URL**.
3. In the **Confirmation URL** dialog box, click the link to confirm the URL. This does not close this dialog box. If the link is successful, a message is displayed that confirms the subscription is active.

> **Note:**
>
> The **Confirmation URL** link expires after a few minutes. You must make sure that you click the link before it expires.

4. Return to the **Confirmation URL** dialog box and select the **Check only when the cloud store confirmation process is complete** check box, and click **OK**.

Once you finish the above steps, upload a new file to the Microsoft Azure container you created in Step 4: Create a container.

1. Navigate to the container you created.
2. Select the **Container** to show a list of blobs it contains.
3. Select **Upload** button to open your local repository and browse the file you need to upload as a block blob.
4. Select the **Upload** button to upload the blob.
5. You can now view the new blob listed within the container.
6. Return to the Database Actions Live Feeds page and find the card for the live table feed you are configuring for a notification-based feed.
7. Click the **Actions** (three vertical dots) icon on the card, and select **Live table feed Run Details**.

You should be able to view logs for the blob uploaded to the Live Feed table from the Microsoft Azure storage in **Live table feed Run Details** window.

For more details on how to create a topic and subscription in Azure portal, refer to Azure Event Grid Notifications.

Transform Data Using Data Studio

Data Transforms is an easy-to-use graphical user interface that you can use to design graphical data transformations for data integration with Autonomous Database.

- The Data Transforms Page
  Data Transforms is an easy-to-use graphical user interface that Oracle Autonomous Database users can use to design graphical data transformations for data integration.

The Data Transforms Page

Data Transforms is an easy-to-use graphical user interface that Oracle Autonomous Database users can use to design graphical data transformations for data integration.

Data Transforms allows you to design data transformations in the form of data loads, data flows, and workflows, without requiring you to write any code. Data loads provide a convenient way of loading data into Autonomous Database, data flows define how the data is moved and transformed between different systems, while the workflows define the sequence in which the data flows are executed.

> **Note:**
>
> If you do not see the Data Transforms card then your database user is missing the required DATA_TRANSFORM_USER role.

After your data flows and workflows are ready, you can execute the mappings immediately or schedule to execute them at a later time. Oracle Data Transforms run-time agent orchestrates the execution of jobs. On execution, Oracle Data Transforms generates the code for you.

You can launch Data Transforms in any of the following ways:

- **Oracle Cloud Marketplace:** Create a Data Transforms instance from Oracle Cloud Marketplace. Data Transforms is available as a separate listing on Marketplace called Data Integrator: Web Edition.

- **Database Actions Data Studio:** Navigate to Database Actions Data Studio Page, and click **Data Transforms** in the Database Actions page.
  If you have already registered a Data Transforms instance from OCI Marketplace with the Autonomous Database, the **Data Transforms** card on the Database Actions page will continue to take you to your Marketplace instance.

Access to the standard set of Data Transforms features may depend on where you launch Data Transforms from. In this documentation, certain topics could include any of the following badges to indicate features that may or may not be available for use:

- APPLIES TO: ✔ Data Transforms that is available as a separate listing on Marketplace called Data Integrator: Web Edition.
- APPLIES TO: ✔ Data Transforms instance that is registered with Autonomous Database.
- APPLIES TO: ✔ Data Transforms that is part of the suite of data tools built into Oracle Autonomous Database.

- Access Oracle Data Transforms From Data Studio
- Data Transforms Notes
- Work with Connections
  Connections help you to connect Data Transforms to various technologies reachable from your OCI network.
- Create a Data Load
  A data load allows you to load multiple data entities from a source connection to a target connection.
- Run a Data Load
  After you create the data load, you are taken to the Data Load Detail page that displays the details that you need to run a data load.
- Work with Data Entities
  A Data Entity is a tabular representation of a data structure.
- Work with Projects
  A project is the top-level container, which can include multiple folders to organize your data flows or work flows into logical groups.
- Create a Data Flow
  A data flow defines how the data is moved and transformed between different systems.
- Schedule Data Flows or Workflows
  You can schedule workflows and data flows to run at specified time interval.
- Monitor Status of Data Loads, Data Flows, and Workflows
  When you run a data load, data flow, or workflow Oracle Data Transforms runs jobs in the background to complete the request. You can view the status of the job in the panel on the bottom right of the Data Load Details, the Data Flow Editor, and the Workflow Editor page.
- Introduction to Workflows
  A workflow is made up of multiple flows organized in a sequence in which they must be executed.
- Create and Manage Jobs
  When you execute a data load, data flow or workflow Oracle Data Transforms creates a job to complete the process in the background.

Access Oracle Data Transforms From Data Studio

APPLIES TO: ✔ Data Transforms that is part of the suite of data tools built into Oracle Autonomous Database.

Data Transforms combines all the elements of data integration - data movement, data synchronization, data quality, and data management to ensure that information is timely, accurate, and consistent across complex systems.

To use the Data Transforms tool you must access Database Actions as the ADMIN user or have the DATA_TRANSFORM_USER role assigned. See Manage User Profiles with Autonomous Database for information on granting roles.

To access the Data Transforms tool:

1. Login as an ADMIN user or a user with the DATA_TRANSFORM_USER role assigned.
2. Click **Data Transforms** in the Database Actions page, or click the Selector icon and select **Data Transforms** from the Data Studio menu in the navigation pane.

> **Note:**
>
> Data Transforms is also available as a separate listing on OCI Marketplace. If you have already registered a Data Transforms instance from OCI Marketplace with the Autonomous Database, the Data Transforms card on the Database Actions page will continue to take you to your Marketplace instance. If you wish to use the embedded Data Transforms, then you must unregister the Marketplace instance. See Unregister the ODI Instance from Autonomous Database.

3. When you login to the Data Transforms tool for the first time, you need to provide the database user credentials to sign in.
   It takes approximately 1-2 minutes for the Data Transforms service to start. After the service starts, the Data Transforms home page opens.
4. From the left pane of the Home page, click the **Connections** tab to view the newly created Autonomous Database connection.
   4. Click the **Actions** icon next to the connection and select **Edit**.
5. In the **Update Connection** page, enter the database username and password to use the connection. Click **Update** to save the changes.

Data Transforms Notes

APPLIES TO: ● Data Transforms that is part of the suite of data tools built into Oracle Autonomous Database.

Notes for using Data Transforms.

- **Enable access to private data sources from Autonomous Database**: If your Autonomous Database is configured to use a Private Endpoint, then you can only access private data sources from clients in the same Virtual Cloud Network (VCN). See Configuring Network Access with Private Endpoints for more information.

  Data Transforms jobs stuck in the Running status: If there are any Data Transforms jobs that are stuck in the Running status for an interminably long time, either stop the job immediately or delete the job and then rerun it. You may want to do this to avoid unwanted usage of resources for your tenancy. If the issue persists, file a service request at Oracle Cloud Support or contact your support representative.

Work with Connections

Connections help you to connect Data Transforms to various technologies reachable from your OCI network.

This section describes the generic steps to create a connection. The displayed connection detail options may vary depending on the selected connection type.

Apart from the connection types listed in Supported Connection Types you can create custom connectors, which you can use to connect Data Transforms to any JDBC supported data sources. See Create Custom Connectors.

To create a new connection:

1. From the left pane of the Home page, click the **Connections** tab.

   Connections page appears.
2. Click **Create Connection**.

   Create Connection page slides in.
3. Do one of the following:
   - In the **Select Type** field, enter the name or part of the name of the connection type.
   - Select the type of connection that you wish to create.
     - Databases - Allows you to configure any connection type for supported database types.
     - Applications - Allows you to configure any connection type for supported applications.
     - Services - Allows you to configure any connection type for supported services.
4. After selecting the required connection type, click **Next**.
5. The **Connection Name** field is pre-populated with a default name. You can edit this value.
6. For **Connection Details**, provide the connection details for the selected type such as:
   - Connection -
     - JDBC URL - The URL to connect to the data server. For example:
       `jdbc:weblogic:sqlserver://hostname:port[;property=value[;...]]`

       For connectors that use an autoREST driver that provides the model files along with the driver, specify the servername and other properties required to connect to that datasource. For example:

       `jdbc:weblogic:autorest://servername;[property=value[;...]]`
     - User - The user name, if required, for connecting to the server.
     - Password - The password for connecting to server.
   - Advanced Options
     - Array Fetch Size - When reading large volumes of data from a data server, Oracle Data Transforms fetches successive batches of records. This value is the number of rows (records read) requested by Oracle Data Transforms on each communication with the data server.
     - Batch Update Size - When writing large volumes of data into a data server, Oracle Data Transforms pushes successive batches of records. This value is the number of rows (records written) in a single Oracle Data Transforms INSERT command.

– Degree of Parallelism for Target - This value indicates the number of threads allowed for a loading task. The default value is 1. The maximum number of threads allowed is 99.

> **Note:**
>
> Connection details are specific and the above options vary based on the selected connection type.

7. After providing all the required connection details, click **Test Connection** to test the connection.

   If the test connection fails, check whether the Autonomous Database from where you are accessing Data Transforms is configured to use a private endpoint.

8. Click **Create**.

   The new connection is created.

The newly created connections are displayed in the **Connections** page.

Click the **Actions** icon (⋮) next to the selected connection to perform the following operations:

- Select **Edit** to edit the provided connection details.
- Select **Test Connection** to test the created connection.
- Select **Delete Schema** to delete schemas.
- Select **Delete Connection** to delete the created connection.

You can also search for the required connection to know its details based on the following filters:

- **Name** of the connection.
- **Technology** associated with the created connection.
- Supported Connection Types
  This topic lists the connection types that are supported for connecting to Data Transforms.
- Create Custom Connectors
- Create a Data Transforms Connection for Remote Data Load
  You can connect to an existing Data Transforms instance and run a data load remotely.
- Create a Delta Share Connection
  Databricks Delta Share is an open protocol for secure data sharing. Oracle Data Transforms integrates with Delta Share to load data to Oracle Autonomous Database. You can use the Delta Share connection to load data from Databricks or Oracle Data Share.
- Create an Oracle Business Intelligence Cloud Connector Connection
  Oracle Business Intelligence Cloud Connector (BICC) allows you to extract business data from a data source and load it into configured external storage.
- Create an Oracle Financials Cloud Connection
  You can fetch real time transactional data from Oracle Financials Cloud REST endpoints, import the data entities into Data Transforms, and use them as a source in a data flow.

- Create and Use an Oracle NetSuite Connection
  Data Transforms uses the Oracle NetSuite JDBC Driver to connect to the Oracle NetSuite application. For Oracle NetSuite connections, Data Transforms allows you to load pre-built dataflows and workflows that you can run to transfer data from NetSuite to your target schema.
- Create an Oracle Object Storage Connection
  You can use Data Transforms for uploading data from Oracle Object Storage to Autonomous Database.
- Create a REST Server Connection

Supported Connection Types

This topic lists the connection types that are supported for connecting to Data Transforms.

> **Note:**
>
> APPLIES TO: Data Transforms that is available as a separate listing on Marketplace called Data Integrator: Web Edition.
>
> - For the connectors that require driver installation, you need to copy the jar files to the `/u01/oracle/transforms_home/userlibs` directory before you add the connection.
> - Apart from the connection types listed here you can create custom connectors, which you can use to connect Data Transforms to any JDBC supported data sources. See Create Custom Connectors.

Name	Type	Supported in Data Integrator: Web Edition	Supported in Data Transforms built into Autonomous Database	Supported in Data Transforms built into OCI GoldenGate	Supports Write Operation	Notes
Aha!	Application	Yes	Yes	Yes	No	
Ahrefs	Application	Yes	Yes	Yes	No	
Amazon Aurora	Database	Yes	Yes	Yes	Yes	
Amazon EMR Hive	Database	Yes	Yes	Yes	No	
Amazon Redshift	Database	Yes	Yes	Yes	Yes	
Apache Hive	Database	Yes	Yes	Yes	No	
Apache Spark SQL	Database	Yes	Yes	Yes	No	
AWS S3	Database	Yes	Yes	Yes	No	
Azure Billing	Application	Yes	Yes	Yes	No	
Azure Compute	Database	Yes	Yes	Yes	No	
Azure Data Lake Storage	Database	Yes	Yes	Yes	No	

Name	Type	Supported in Data Integrator: Web Edition	Supported in Data Transforms built into Autonomous Database	Supported in Data Transforms built into OCI GoldenGate	Supports Write Operation	Notes
Azure Reserved VM Instances	Database	Yes	Yes	Yes	No	
Azure Resource Health	Application	Yes	Yes	Yes	No	
Azure SQL Database	Database	Yes	Yes	Yes	Yes	
Azure Synapse Analytics	Database	Yes	Yes	Yes	Yes	
BigCommerce	Application	Yes	Yes	Yes	No	Requires driver installation
Cassandra	Database	Yes	Yes	Yes	Yes	
Cloudera CDH Hive	Database	Yes	Yes	Yes	No	
Cloudera Impala	Database	Yes	Yes	Yes	No	
Confluence Cloud	Database	Yes	Yes	Yes	No	
Data Transforms	Service	Yes	Yes	Yes	No	For instructions on connecting to an existing Data Transforms instance, see Create a Data Transforms Connection for Remote Data Load.
DataStax	Application	Yes	Yes	Yes	Yes	
Delta Share	Application	Yes	Yes	Yes	No	For instructions on creating a connection using Delta Share, see Create a Delta Share Connection
DocuSign	Database	Yes	Yes	Yes	No	
eBay	Application	Yes	Yes	Yes	No	Requires driver installation
EnterpriseDB	Database	Yes	Yes	Yes	Yes	

Name	Type	Supported in Data Integrator: Web Edition	Supported in Data Transforms built into Autonomous Database	Supported in Data Transforms built into OCI GoldenGate	Supports Write Operation	Notes
FinancialForce	Application	Yes	Yes	Yes	Yes	
FourSquare	Application	Yes	Yes	Yes	No	
Generic Rest	Application	Yes	Yes	Yes	No	For information about connecting to any REST service endpoint to create a connection, see Create a REST Server Connection.
Generic Rest Config	Application	Yes	No	No	No	For information about connecting to any REST service endpoint to create a connection, see Create a REST Server Connection.
GitHub	Application	Yes	Yes	Yes	No	
Google Ads	Application	Yes	No	No	Depends on the driver	Requires driver installation
Google AdSense	Application	Yes	Yes	Yes	No	
Google Analytics	Application	Yes	Yes	Yes	No	
Google BigQuery	Database	Yes	Yes	Yes	No	
Google Calendar	Application	Yes	Yes	Yes	No	
Google Campaign Manager	Application	Yes	Yes	Yes	No	
Google Contacts	Application	Yes	Yes	Yes	No	
Google Drive	Database	Yes	Yes	Yes	No	
Google Search Ads 360	Application	Yes	Yes	Yes	No	
Greenplum	Database	Yes	Yes	Yes	No	

Name	Type	Supported in Data Integrator: Web Edition	Supported in Data Transforms built into Autonomous Database	Supported in Data Transforms built into OCI GoldenGate	Supports Write Operation	Notes
Hortonworks Hive	Database	Yes	Yes	Yes	No	
HubSpot	Application	Yes	Yes	Yes	No	
Hypersonic SQL	Database	Yes	Yes	Yes	Yes	
IBM BigInsights	Database	Yes	Yes	Yes	No	
IBM DB2 Hosted	Database	Yes	Yes	Yes	Yes	
IBM DB2 UDB	Database	Yes	Yes	Yes	Yes	
IBM DB2 Warehouse	Database	Yes	Yes	Yes	Yes	
IBM DB2/400	Database	Yes	Yes	Yes	Yes	
Informix	Database	Yes	Yes	Yes	No	
Jira	Application	Yes	Yes	Yes	No	
Klaviyo	Application	Yes	Yes	Yes	No	
Magento	Application	Yes	No	No	Depends on the driver	Requires driver installation
Mailchimp	Application	Yes	Yes	Yes	No	
MapR Hive	Database	Yes	Yes	Yes	No	
Marketo	Application	Yes	Yes	Yes	No	
Microsoft Dynamics 365	Application	Yes	Yes	Yes	Yes	
Microsoft SharePoint	Application	Yes	Yes	Yes	Yes	
Microsoft SQL Server	Database	Yes	Yes	Yes	Yes	
Mongo DB	Database	Yes	Yes	Yes	Yes	

Name	Type	Supported in Data Integrator: Web Edition	Supported in Data Transforms built into Autonomous Database	Supported in Data Transforms built into OCI GoldenGate	Supports Write Operation	Notes
MySQL	Database	Yes	Yes	Yes	Yes	Make sure that the system variable property `sql_require_primary_key` is set to OFF. Otherwise, an ADW to MySQL mapping could fail with a “Table does not exist” error.
MySQL Heatwave	Database	Yes	Yes	Yes	Yes	Make sure that the system variable property `sql_require_primary_key` is set to OFF. Otherwise, an ADW to MySQL Heatwave mapping could fail with a “Table does not exist” error.

Name	Type	Supported in Data Integrator: Web Edition	Supported in Data Transforms built into Autonomous Database	Supported in Data Transforms built into OCI GoldenGate	Supports Write Operation	Notes
Netezza	Database	Yes	No	No	Depends on the driver	Oracle Data Transforms uses the Netezza JDBC to connect to a NCR Netezza database. This driver must be installed in your Data Transforms `userlibs` directory. See Download the Netezza JDBC driver for more information.
Oracle	Database	Yes	Yes	Yes	Yes	For Data Integrator Web Edition, write operation is supported only on Oracle cloud database targets. For details refer to the Oracle terms of use before deploying the image from OCI marketplace.
Oracle Analytics Cloud	Application	Yes	Yes	Yes	No	

Name	Type	Supported in Data Integrator: Web Edition	Supported in Data Transforms built into Autonomous Database	Supported in Data Transforms built into OCI GoldenGate	Supports Write Operation	Notes
Oracle Business Intelligence Cloud (BICC) Connector	Application	Yes	Yes	Yes	No	For information about creating a connection using Oracle Business Intelligence Cloud (BICC) Connector, see Create an Oracle Business Intelligence Cloud Connector Connection.
Oracle EBS	Application	Yes	Yes	Yes	Yes	
Oracle Financials Cloud	Application	Yes	Yes	Yes	No	For information about creating a connection using Oracle Financials Cloud, see Create an Oracle Financials Cloud Connection.
Oracle Fusion ERP	Application	Yes	Yes	Yes	No	
Oracle Fusion Sales	Application	Yes	Yes	Yes	No	
Oracle Fusion Service	Application	Yes	Yes	Yes	No	
Oracle GoldenGate – OCI	Service	Yes	Yes	Yes	Yes	
Oracle Marketing Cloud	Application	Yes	Yes	Yes	Yes	

Name	Type	Supported in Data Integrator: Web Edition	Supported in Data Transforms built into Autonomous Database	Supported in Data Transforms built into OCI GoldenGate	Supports Write Operation	Notes
Oracle NetSuite	Application	Yes	Yes	Yes	No	For information about creating a connection using Oracle Netsuite, see Create and Use an Oracle NetSuite Connection.
Oracle Object Storage	Database	Yes	Yes	Yes	Yes	For information about creating a connection using Oracle Object Storage, see Create an Oracle Object Storage Connection.
Oracle People Soft	Application	Yes	Yes	Yes	No	
Oracle Sales Cloud	Application	Yes	Yes	Yes	No	
Oracle Service Cloud	Application	Yes	Yes	Yes	No	
Oracle SIEBEL	Application	Yes	Yes	Yes	No	
PayPal	Application	Yes	Yes	Yes	No	
Pivotal HD	Database	Yes	Yes	Yes	No	
Pivotal HDB	Database	Yes	Yes	Yes	No	
PostgreSQL	Database	Yes	Yes	Yes	Yes	
Qmetry	Application	Yes	Yes	Yes	No	
QuickBooks Online	Application	Yes	Yes	Yes	No	
QuickBooks Payments	Application	Yes	Yes	Yes	No	
Quora Ads	Application	Yes	Yes	Yes	No	
Sage	Application	Yes	Yes	Yes	No	
Salesforce Chatter	Application	Yes	Yes	Yes	No	
Salesforce.com	Application	Yes	Yes	Yes	Yes	

Name	Type	Supported in Data Integrator: Web Edition	Supported in Data Transforms built into Autonomous Database	Supported in Data Transforms built into OCI GoldenGate	Supports Write Operation	Notes
SAP BW/ 4HANA	Database	Yes	Yes	Yes	No	
SAP HANA	Application	Yes	Yes	Yes	No	
SAP NetWeaver	Database	Yes	Yes	Yes	No	
SAP S/ 4HANA Cloud	Application	Yes	Yes	Yes	No	
Semrush	Application	Yes	Yes	Yes	No	
ServiceNow	Service	Yes	Yes	Yes	No	
Shopify	Application	Yes	Yes	Yes	No	Requires driver installation
Snowflake	Database	Yes	Yes	Yes	Yes	
Square	Application	Yes	Yes	Yes	No	
Stripe	Application	Yes	Yes	Yes	No	
Sybase As Anywhere	Database	Yes	Yes	Yes	Yes	
Sybase as Enterprise	Database	Yes	Yes	Yes	Yes	
Sybase AS IQ	Database	Yes	Yes	Yes	Yes	
TeamCity	Application	Yes	Yes	Yes	No	

Name	Type	Supported in Data Integrator: Web Edition	Supported in Data Transforms built into Autonomous Database	Supported in Data Transforms built into OCI GoldenGate	Supports Write Operation	Notes
Teradata	Database	Yes	No	No	Depends on the driver	Data Transforms uses the Teradata JDBC Driver to connect to a Teradata Database. To use Teradata as a data source the Teradata Gateway for JDBC must be running, and this driver must be installed in your Data Transforms `userlibs` directory. You will find the driver here: https://downloads.teradata.com/download/connectivity/jdbc-driver.
Tumblr	Application	Yes	Yes	Yes	No	
Twitter	Application	Yes	Yes	Yes	No	
Veeva CRM	Application	Yes	Yes	Yes	Yes	
Volusion	Application	Yes	Yes	Yes	No	
Wistia	Application	Yes	Yes	Yes	No	
WooCommerce	Application	Yes	No	No	Depends on the driver	Requires driver installation
WordPress	Application	Yes	Yes	Yes	No	
Workday	Application	Yes	No	No	Depends on the driver	Requires driver installation
Xero	Application	Yes	Yes	Yes	No	Requires driver installation
Yelp	Application	Yes	Yes	Yes	No	
Zendesk	Application	Yes	Yes	Yes	No	Requires driver installation
Zoho CRM	Application	Yes	Yes	Yes	No	

Name	Type	Supported in Data Integrator: Web Edition	Supported in Data Transforms built into Autonomous Database	Supported in Data Transforms built into OCI GoldenGate	Supports Write Operation	Notes
Zoom	Application	Yes	Yes	Yes	No	

Create Custom Connectors

APPLIES TO: Data Transforms that is available as a separate listing on Marketplace called Data Integrator: Web Edition.

The Custom Connections page of the Administration tab of Oracle Data Transforms helps you to create custom connectors that point to any JDBC supported data sources.

The custom connectors will be listed in the Create Connection page where you can use them to connect data sources to Data Transforms. See Work with Connections for more information. To create a new connector:

1. In the left pane, click **Administration**.
   A warning message appears.
2. Click **Continue**.
3. In the left pane, click **Custom Connections**.
   Custom Connections screen appears.
4. Click **Create Connection Type**.
   The Create Connection Type page appears.
5. From the **Category** drop-down select the type of connection that you wish to create whether database, application, or service.
6. Enter a name for the connection.
7. Enter the name of the JDBC Driver.
8. Click **OK**.

The newly created custom connection appears in the list and are available in the Create Connection page.

Click the **Actions** icon (⋮) next to the selected connection to perform the following operations:

- Select **Edit**, to edit the provided connection details.
- Select **Delete**, to delete the created connection.

> **Note:**
>
> You cannot delete custom connectors that have existing connections.

Create a Data Transforms Connection for Remote Data Load

You can connect to an existing Data Transforms instance and run a data load remotely.

To create this connection, you need to specify the URL of the Data Transforms instance along with the name of the ODI rest API from where you want to run the data load.

To define a Data Transforms connection:

1. From the left pane of the Home page, click the **Connections** tab.

   Connections page appears.
2. Click **Create Connection**.

   Create Connection page slides in.
3. For **Select Type**,
   - In the **Name** field, enter the name of the newly created connection
   - Select **Services** as the type of connection that you wish to create.
4. In the **Endpoint URL** textbox, enter the URL of the ODI rest API from where you want to run the data load. Enter the URL in the format `http://<host-ip-address>:<port>/odi-rest`.
5. In the **User** text box enter SUPERVISOR as the user name.
6. In the **Password** text box enter the ODI Supervisor password.
7. After providing all the required connection details, click **Test Connection** to test the established connection.
8. Click **Create**.

   The new connection is created.

The newly created connections are displayed in the **Connections** page.

Click the **Actions** icon (⋮) next to the selected connection to perform the following operations:

- Select **Edit** to edit the provided connection details.
- Select **Test Connection** to test the created connection.
- Select **Delete Schema** to delete schemas.
- Select **Delete Connection** to delete the created connection.

You can also search for the required connection to know its details based on the following filters:

- **Name** of the Connection.
- **Technology** associated with the created Connection.

Create a Delta Share Connection

Databricks Delta Share is an open protocol for secure data sharing. Oracle Data Transforms integrates with Delta Share to load data to Oracle Autonomous Database. You can use the Delta Share connection to load data from Databricks or Oracle Data Share.

To use Databricks as a source, you need to specify the URL of the Delta Sharing server along with the bearer token that lets you access the Delta Lake share server. To use Oracle Data

Share as a source, you need to specify the URL for the token end point along with a client ID and the secret key.

This topic has the following sections:

- Creating the Delta Share Connection
- Creating and Running a Delta Share Data Load

Creating the Delta Share Connection

To define a Delta Share connection:

1. From the left pane of the Home page, click the **Connections** tab.
   Connections page appears.
2. Click **Create Connection**.
   Create Connection page slides in.
3. Do one of the following:
   - In the **Select Type** field, enter the name or part of the name of the connection type.
   - Select the **Databases** tab.
4. Select **Delta Share** as the connection type.
5. Click **Next**.
6. The **Connection Name** field is pre-populated with a default name. You can edit this value.
7. In the **Share Endpoint URL** textbox, enter the URL of the Delta Sharing server. Enter the value in the `<host>:<port>/<shareEndpoint>/` format.
8. In the Connection section, do one of the following:
   - Select **Oracle Data Share** and provide the **Token Endpoint URL**, **Client ID**, and **Client Secret** for accessing the share.
     You can get this information from the Delta Share Profile JSON document that you will need to download from supplied to you by the Share Provider. (This is also where they get the Share Endpoint URL from)

     You can get this information from the Delta Share Profile JSON document that you can download from the activation link that is provided by the Data Share provider to access their share.
   - Select **Databricks** and in the **Bearer Token** text box enter the token for connecting to the Delta Sharing server.
9. If you need to use a proxy to access the Delta Share Server or Delta Share Storage configure the following settings:
   - In the **Proxy Host** textbox, enter the host name of the proxy server to be used for the connection.
   - In the **Proxy Port** textbox, enter the port number of the proxy server.
   - Select the following checkboxes depending on where the proxy is required:
     - **Use Proxy to access Delta Share Server**
     - **Use Proxy to access Delta Share Storage**
10. Click **Test Connection**, to test the established connection.
11. After providing all the required connection details, click **Create**.
    The new connection is created.

The newly created connections are displayed in the **Connections** page.

Click the **Actions** icon (⋮) next to the selected connection to perform the following operations:

- Select **Edit**, to edit the provided connection details.
- Select **Test Connection**, to test the created connection.
- Select **Delete Schema**, to delete schemas.
- Select **Delete Connection**, to delete the created connection.

You can also search for the required Connection to know its details based on the following filters:

- **Name** of the Connection.
- **Technology** associated with the created Connection.

Creating and Running a Delta Share Data Load

To load data from Delta Share into Oracle Autonomous Database, the Oracle connection user must be an Admin user. Admin privileges are required so that the Oracle user can create and insert data into tables in another schema.

When you run the data load, Data Transforms loads the data onto a corresponding table in the target schema. The data load runs incrementally. The very first time you run a data load, Data Transforms copies all the data into new tables. For every subsequent data load run, it only uploads the changes. Any additions or deletions in the records will reflect in the target tables. Note that if there is any metadata change in the table, for example a column is added, Data Transforms creates a new table to load the data on to the target server. You could create a workflow, add the data load as a step, create a schedule to run the workflows at a predefined time interval. See

To create and run a Delta Share data load:

1. Do one of the following:
   - On the Home page, click **Load Data**. The Create Data Load wizard appears.
     In the Create Data Load tab, enter a name if you want to replace the default value, add a description, and select a project from the drop-down.
   - On the Home page, click **Projects**, and then the required project tile. In the left pane, click **Data Loads**, and then click **Create Data Load**. The Create Data Load wizard appears.
     In the Create Data Load tab, enter a name if you want to replace the default value and add a description.
2. Click **Next**.
3. In the Source Connection tab,
   a. From the **Connection Type** drop-down, select **Delta Share**.
   b. from the **Connection** drop-down, select the required connection from which you wish to add the data entities.
   c. Select the share that you want to load tables from the **Share** drop-down. The drop-down lists all the shares for the selected connection.
   d. Click **Next**.
4. In the Target Connection tab,
   a. From the **Connection Type** drop-down, select **Oracle** as the connection type.

> **Note:**
>
> This drop-down lists only JDBC type connections.

b. From the **Connection** drop-down, select the required connection from to you wish to load the data entities.

c. Enter a unique name in the **Schema** textbox.

d. Click **Save**.

The Data Load Detail page appears listing all the tables in the selected share with their schema names.

> **Note:**
>
> For Delta Share data loads the Data Load Detail page only includes the option. You cannot apply different actions - incremental merge, incremental append, recreate, truncate, append - on the data entities before loading it to the target schema. This is to make sure that the data is consistent between the Delta Sharing server and the target schema.

5. Click to run the data load.
   A confirmation prompt appears when the data load starts successfully.

To check the status of the data load, see the Status panel on the right below the Target Schema details. For details about the Status panel, see Monitor Status of Data Loads, Data Flows, and Workflows. This panel shows links to the jobs that execute to run this data load. Click the link to monitor the progress on the Job Details page. For more information about jobs, see Create and Manage Jobs.

All the loaded data entities along with their details are listed in the Data Entities page. To view the statistics of the data entities, click the **Actions** icon ( ) next to the data entity, click **Preview**, and then select the **Statistics** tab. See View Statistics of Data Entities for information.

Create an Oracle Business Intelligence Cloud Connector Connection

Oracle Business Intelligence Cloud Connector (BICC) allows you to extract business data from a data source and load it into configured external storage.

To create an Oracle BICC connection you need to first configure external storage using the OCI Object Storage Connection tab in the BICC Console. You need to specify these connection details when you define the connection in Oracle Data Transforms.

You can use the BICC connection to choose the offerings whose data stores you want to extract. Data Transforms uses an Oracle Object Storage Data Server used by Oracle BICC to stage the extracted files, which you can then use as a source for mapping. Note that you cannot use an Oracle BICC connection as a target for mapping.

To define an Oracle BICC connection,

1. From the left pane of the Home page, click the **Connections** tab.

   Connections page appears.

2. Click **Create Connection**.

   Create Connection page slides in.

3. Do one of the following:
   - In the **Select Type** field, enter the name or part of the name of the connection type.
   - Select the **Applications** tab.
4. Select **Oracle BI Cloud Connector** as the connection type.
5. Click **Next**.
6. The **Connection Name** field is pre-populated with a default name. You can edit this value.
7. Enter the URL in the **BI Cloud Connector Service URL** textbox.
8. In the **Connection** section, enter the following details:
   - In the **User** text box enter your Oracle Cloud Infrastructure username.
   - In the **Password** text box enter the auth token.
9. In the **Storage** section, enter the following details:
   - In the **External Storage BICC Name** text box enter the name of the external storage as it appears in the Oracle BI Cloud Connector Console.
   - In the **External Storage Bucket** text box specify the bucket into which extracts are uploaded. Bucket names are obtained in the OCI Console.
   - In the **External Storage Name Space** text box specify the namespace. Namespace is obtained in the OCI Console.
   - In the **External Storage Region** text box enter the region.
   - In the **External Storage User** text box enter the name of the user logging into the Oracle Storage Cloud Service. The value specified in this field must be the same as the User Name configured in the Oracle BI Cloud Connector Console.
   - In the **External Storage Token** text box enter the password of the logged in user. The value specified in this field must be the same as the Password configured in the Oracle BI Cloud Connector Console.
10. Click **Test Connection** to test the established connection.
11. Click **Create**.

    The new connection is created.

The newly created connections are displayed in the **Connections** page.

Click the **Actions** icon (⋮) next to the selected connection to perform the following operations:

- Select **Edit**, to edit the provided connection details.
- Select **Test Connection**, to test the created connection.
- Select **Delete Schema**, to delete schemas.
- Select **Delete Connection**, to delete the created connection.

You can also search for the required Connection to know its details based on the following filters:

- **Name** of the Connection.

- **Technology** associated with the created Connection.

Create an Oracle Financials Cloud Connection

You can fetch real time transactional data from Oracle Financials Cloud REST endpoints, import the data entities into Data Transforms, and use them as a source in a data flow.

To create an Oracle Financials Cloud connection you need to choose a temporary schema where Data Transforms can create data entities after the reverse-engineering operation.

To define an Oracle Financials Cloud connection,

1. From the left pane of the Home page, click the **Connections** tab.

   Connections page appears.
2. Click **Create Connection**.

   Create Connection page slides in.
3. Do one of the following:
   - In the **Select Type** field, enter the name or part of the name of the connection type.
   - Select the **Applications** tab.
4. Select **Oracle Financials Cloud** as the connection type.
5. Click **Next**.
6. The **Connection Name** field is pre-populated with a default name. You can edit this value.
7. In the **REST Service URL** textbox, enter the URL of the endpoint that services the REST resources.
8. In the **Proxy Host** textbox, enter the host name of the proxy server to be used for the connection.
9. In the **Proxy Port** textbox, enter the port number of the proxy server.
10. In the **User** text box enter the user name for connecting to the REST endpoint.
11. In the **Password** text box enter the password for connecting to the REST endpoint.
12. Choose a connection from the **Staging Connection** drop-down list. The list displays only existing Autonomous Database connections. To use a different connection, create the connection before you reach this page.
13. After providing all the required connection details, click **Create**.

    The new connection is created.
14. Click **Test Connection**, to test the established connection.

The newly created connections are displayed in the **Connections** page.

Click the **Actions** icon (⋮) next to the selected connection to perform the following operations:

- Select **Edit**, to edit the provided connection details.
- Select **Test Connection**, to test the created connection.
- Select **Delete Schema**, to delete schemas.
- Select **Delete Connection**, to delete the created connection.

You can also search for the required Connection to know its details based on the following filters:

- **Name** of the Connection.
- **Technology** associated with the created Connection.

Create and Use an Oracle NetSuite Connection

Data Transforms uses the Oracle NetSuite JDBC Driver to connect to the Oracle NetSuite application. For Oracle NetSuite connections, Data Transforms allows you to load pre-built dataflows and workflows that you can run to transfer data from NetSuite to your target schema.

This topic has the following sections:

Creating the Oracle NetSuite Connection

To define an Oracle NetSuite connection:

1. From the left pane of the Home page, click the **Connections** tab.
   Connections page appears.
2. Click **Create Connection**.
   Create Connection page slides in.
3. Do one of the following:
   - In the **Select Type** field, enter the name or part of the name of the connection type.
   - Select the **Applications** tab.
4. Select **Oracle NetSuite** as the connection type.
5. Click **Next**.
6. The **Connection Name** field is pre-populated with a default name. You can edit this value.
7. In the **JDBC URL** textbox, enter the URL of the SuiteAnalytics Connect server to be used for the connection.
8. In the **User** text box enter the user name for connecting to the data server.
9. In the **Password** text box enter the password for connecting to the data server.
10. In the **Account ID** textbox, enter the account ID for connecting to the data server.
11. In the **Role ID** textbox, enter the role ID for connecting to the data server.
12. If the source that you want to use for mapping is a saved search, you need to also specify the following details in **Saved Search Extraction**:
    - **Application ID**: Enter the NetSuite Application ID for Data Transforms.
    - **Version**: Enter the NetSuite version number.
13. Select the checkbox in **Build Data Model** to install pre-built dataflows and workflows that you can run to extract data from NetSuite and move it to your Oracle target schema using the Build Data Warehouse wizard.
14. Click **Test Connection**, to test the established connection.
15. After providing all the required connection details, click **Create**.
    The new connection is created.

The newly created connections are displayed in the **Connections** page.

Click the **Actions** icon (⋮) next to the selected connection to perform the following operations:

- Select **Edit**, to edit the provided connection details.
- Select **Test Connection**, to test the created connection.
- Select **Build Data Warehouse**, to select the functional areas and create the NetSuite Data Warehouse in the target schema. See Using the Build Data Warehouse Wizard for more information.
- Select **Delete Schema**, to delete schemas.
- Select **Delete Connection**, to delete the created connection.

You can also search for the required Connection to know its details based on the following filters:

- **Name** of the Connection.
- **Technology** associated with the created Connection.

Using the Build Data Warehouse Wizard

Data in your NetSuite account is grouped into business or subject areas in the Analytics Warehouse. The Build Data Warehouse wizard allows you to select the areas that you want to include in the newly created Data Warehouse.

To use the Build Data Warehouse Wizard:

1. On the Home page, click the **Connections** tab. The **Connections** page appears.
2. Click the Actions icon (⋮) next to the Oracle NetSuite connection that you want to use to build the data warehouse and click **Build Data Warehouse**.
   The Build Data Warehouse wizard opens.
3. From the **Connection** drop-down list, choose the Autonomous Database connection where your target schema resides.
4. From the **Staging Schema** drop-down, all schema corresponding to the selected connection are listed in two groups:
   - Existing Schema (ones that you've imported into Oracle Data Transforms) and
   - New Database Schema (ones that you've not yet imported).

   Select the schema that you want to use from the drop-down.
5. Similarly select the **Target Schema**.
6. Click **Next**.
7. Select the **NetSuite Business Areas** that you want to use to transfer data from the NetSuite Data Warehouse to the target schema.
8. Click **Save**.
   Data Transforms starts the process to build the data warehouse. Click **Jobs** on the left pane of the Home page to monitor the progress of the process. When the job completes successfully, Data Transforms creates a Project folder that includes all the pre-built workflows and dataflows, which you can run to transfer data from the NetSuite connection to your target schema. See Running the Pre-Built Workflows to Load Data into the Target Schema for more information.

Running the Pre-Built Workflows to Load Data into the Target Schema

When the Build Data Warehouse wizard completes successfully, Data Transforms creates a project that includes all the pre-built data flows and workflows that you can run to extract data from a Netsuite connection and load it into your target schema.

To view and run the pre-built workflows:

1. Click **Projects** on the left pane of the Home page and select the newly created NetSuite project.
2. Click **Workflows** in the left pane. The following pre-built workflows are listed in the Project Details page:
   - Stage NetSuite Source to SDS
   - Extract Transaction Primary Keys
   - Load SDS to Warehouse
   - Apply Deletes
   - All Workflows
3. Click the Actions icon (⋮) next to the workflow you want to run and click **Start**. Oracle recommends that you run **All Workflows** to execute all the pre-built workflows.

   To see the status of the workflow, click **Jobs** from the left pane in the current project. When the job completes successfully, all the data from the NetSuite connection is loaded into the target schema.

Create an Oracle Object Storage Connection

You can use Data Transforms for uploading data from Oracle Object Storage to Autonomous Database.

To create an Oracle Object Storage connection you need to have an Oracle Cloud Infrastructure username and an auth token. See Getting an Auth Token for information about how to generate the auth token. You need to specify these details when you define the connection in Oracle Data Transforms.

Note the following:

- To use an Oracle Object Storage connection to import data into Data Transforms, you must use a public IP address to access the compute node. If you want to use a private IP address to access the Object Storage service, make sure that you have access to the Internet.
- The supported file format for loading data from Oracle Object Storage to Autonomous Database and vice versa is CSV.
- The supported data types are Numeric, Double, String, and Date.
- Data load is not supported for Oracle Object Storage connections.
- You cannot use an Oracle Object Storage connection as a target for mapping.

To define an Oracle Object Storage connection,

1. From the left pane of the Home page, click the **Connections** tab.

   Connections page appears.
2. Click **Create Connection**.

Create Connection page slides in.

3. Do one of the following:
   - In the **Select Type** field, enter the name or part of the name of the connection type.
   - Select the **Databases** tab.
4. Select **Oracle Object Storage** as the connection type.
5. Click **Next**.
6. The **Connection Name** field is pre-populated with a default name. You can edit this value.
7. Enter the URL in the **Object Storage URL** textbox. You can enter the URL in either of the following formats:
   - URL with fully qualified domain name.
     For example,

     ```
     https://swiftobjectstorage.<your-region>.oraclecloud.com/v1/<your-namespace>/<your-bucket>/<your-file>
     ```

     ```
     https://objectstorage.<your-region>.oraclecloud.com/n/<your-namespace>/b/<your-bucket>/o/<your-file>
     ```

   - If you want to use the URL provided by the OCI Console, specify the URL only till the name of the bucket.
     For example,

     ```
     https://swiftobjectstorage.<your-region>.oraclecloud.com/v1/<your-namespace>/<your-bucket>
     ```

     ```
     https://objectstorage.<your-region>.oraclecloud.com/n/<your-namespace>/b/<your-bucket>/o
     ```

   - If you choose **Credential** as the **Connection Mode** (see step 6), specify the URL till *bucketname*/o
     For example,

     ```
     https://objectstorage.<your-region>.oraclecloud.com/n/<your-namespace>/b/<your-bucket>/o/
     ```

   The values for Region, Namespace and Bucket are auto-populated based on the URL provided.
8. To select the **Connection Mode** do one of the following:
   - Select **Swift Connectivity**, and provide the following details:
     - In the **User Name** text box enter your Oracle Cloud Infrastructure username.
     - In the **Token** text box enter the auth token.
   - Select **Credential** and provide the ODI credential in the **Enter Credential** text box. You must create the credential in the repository and in the Autonomous Database that you created during instance creation. When you create a data flow to map data from Object Storage to Autonomous Database you need to create the ODI credential in target schema as well.

To create the credential, execute the following script:

```
begin DBMS_CLOUD.create_credential( credential_name => '<Credential
name>',
      username => '<oci username>', password => '<auth token>' ); end;
```

9. Click **Create**.

   The new connection is created.

10. Click **Test Connection**, to test the established connection.

The newly created connections are displayed in the **Connections** page.

Click the **Actions** icon (⋮) next to the selected connection to perform the following operations:

- Select **Edit**, to edit the provided connection details.
- Select **Test Connection**, to test the created connection.
- Select **Delete Schema**, to delete schemas.
- Select **Delete Connection**, to delete the created connection.

You can also search for the required Connection to know its details based on the following filters:

- **Name** of the Connection.
- **Technology** associated with the created Connection.

Create a REST Server Connection

You can connect to any REST service endpoint, import the data entities into Data Transforms, and use them as source in a data flow.

To create a generic REST connector, you need to provide the JDBC URL, username, and password to connect to the endpoint. You can also create and upload a config file that contains information such as the authentication methods, endpoints, and tables that you want to import data entities from.

The **Application** tab on the **Create Connection** page includes two connection options to create a generic REST connection - Generic REST and Generic REST Config. This topic has the following sections:

- Creating a Generic REST Connection
- Creating a Generic Rest Connection Using a Config File

The newly created connections are displayed in the **Connections** page.

Click the **Actions** icon (⋮) next to the selected connection to perform the following operations:

- Select **Edit** to edit the provided connection details.
- Select **Test Connection** to test the created connection.
- Select **Delete Schema** to delete schemas.
- Select **Delete Connection** to delete the created connection.

You can also search for the required connection to know its details based on the following filters:

- **Name** of the connection.
- **Technology** associated with the created connection.

Creating a Generic REST Connection

To create this connection you need to specify the REST service URL and choose a temporary schema where Data Transforms can create data entities after the reverse-engineering operation.

To define a REST server connection:

1. From the left pane of the Home page, click the **Connections** tab.
   Connections page appears.
2. Click **Create Connection**.
   Create Connection page slides in.
3. Do one of the following:
   - In the **Select Type** field, enter the name or part of the name of the connection type.
   - Select the **Applications** tab.
4. Select **Generic Rest** as the connection type.
5. Click **Next**.
6. The **Connection Name** field is pre-populated with a default name. You can edit this value.
7. In the **REST Service URL** textbox, enter the URL of the endpoint that services the REST resources.
8. In the **Proxy Host** textbox, enter the host name of the proxy server to be used for the connection.
9. In the **Proxy Port** textbox, enter the port number of the proxy server.
10. In the **User** text box enter the user name for connecting to the REST endpoint.
11. In the **Password** text box enter the password for connecting to the REST endpoint.
12. Choose a connection from the **Staging Connection** drop-down list. The list displays only existing Autonomous Database connections. To use a different connection, create the connection before you reach this page.
13. After providing all the required connection details, click **Test Connection** to test the connection.
14. Click **Create**.
    The new connection is created.

Creating a Generic Rest Connection Using a Config File

APPLIES TO: ✔ Data Transforms that is available as a separate listing on Marketplace called Data Integrator: Web Edition.

To create a generic REST connector, you need the JDBC URL, username, password, and a config file. The config file is a model file with the *file_name.rest* naming convention that you need to upload when you create a REST Server connection. You need to specify the endpoints, table mappings, and the authentication methods to create the config file. You can create the config file using any text editor.

To define a REST server connection using a config file:

1. From the left pane of the Home page, click the **Connections** tab.
   Connections page appears.
2. Click **Create Connection**.
   Create Connection page slides in.
3. Do one of the following:
   - In the **Select Type** field, enter the name or part of the name of the connection type.
   - Select the **Applications** tab.
4. Select **Generic Rest Config** as the connection type.
5. Click **Next**.
6. The **Connection Name** field is pre-populated with a default name. You can edit this value.
7. Use the **Config File** text box to upload the config file that you want to use.
8. In the **JDBC URL** textbox, enter the URL to connect to the server.
9. In the **User** and **Password** text boxes enter the user name and password for connecting to the REST endpoint. You may leave these fields blank if these values are not applicable or are already mentioned in the JDBC URL.
10. After providing all the required connection details, click **Test Connection** to test the connection.
11. Click **Create**.
    The new connection is created.

Create a Data Load

A data load allows you to load multiple data entities from a source connection to a target connection.

You can create a data load in either of the following ways:

- Using the Create Data Load wizard.
- Using the Projects page.

> **Note:**
>
> Data load is not supported for Oracle Object Storage connections.

Creating a Data Load from the Home Page

This section describes the generic steps to create a data load. If you plan to load and transform data using OCI GoldenGate, create the data load using the Projects page. See Creating a Data Load from the Projects Page.

To create a data load from the Home page:

1. From the left pane, click the Home tab. Click **Load Data**.
   The Create Data Load wizard appears.
2. In the Name field, enter a name for the data load. The field is pre-populated with a default name. You can edit this value.
3. Add a description. This is optional.

4. Select a project name from the drop-down. If this your first time here, click the + icon to create a project. If you have logged in as SUPERVISOR, the default project name is `Home`. For other users, the default project name is in the format `<username>_Home`. You can edit the default value. See Work with Projects for more information about projects.
5. Click **Next**.
6. To define your source connection, from the **Connection** drop-down, select the required connection from which you wish to add the data entities. Alternatively, click the + icon to create a new connection. See Work with Connections for more details about connections.
7. In the **Schema** drop-down, all schema corresponding to the selected connection are listed in two groups:
   - Existing Schema (ones that you've imported into Oracle Data Transforms) and
   - New Database Schema (ones that you've not yet imported).

   Select the schema that you want to use from the drop down.

   > **Note:**
   >
   > If there is missing information such as user name or password not specified, wallet missing, and so on, the list may fail to populate with a “This connection has missing information.” error. Click the Edit icon ( ) to open the Update Connection page where you can fill in the missing details.

8. Click **Next**.
9. Similarly, define the target connection.
10. Click **Save**.
    The Data Load Detail page appears listing all the loaded data entities.

Creating a Data Load from the Projects Page

To create a data load from the Projects page,

1. On the Home page, click **Projects**, and then the required project title. In the left pane, click **Data Loads**, and then click **Create Data Load**.
   The Create Data Load wizard appears.
2. In the Name field, enter a name for the data load.
3. Add a description. This is optional.
4. Select the source and target schemas.

   > **Note:**
   >
   > Make sure that you have created connections before you plan to create a data load using the Projects page. See Work with Connections for more details about connections.

   - To use the OCI GoldenGate Deployment Console to load data entities:
     a. Select the **Use GoldenGate** checkbox.
        The Create Data Load page now shows fields that are specific to OCI GoldenGate.

b. Select the GoldenGate connection from the drop-down.

c. To select the source connection, select the Registered Database and the Schema.

d. Similarly, define the target connection.

- To use all other connection types to load entities:

  a. To define your source connection, select the connection from which you wish to add the data entities from the **Connection** drop-down.

  b. From the **Schema** drop-down select the schema that you want to use. All schema corresponding to the selected connection are listed in two groups

     – Existing Schema (ones that you've imported into Oracle Data Transforms) and

     – New Database Schema (ones that you've not yet imported).

5. Click **Create**.
   The Data Load Detail page appears listing all the loaded data entities.

Run a Data Load

After you create the data load, you are taken to the Data Load Detail page that displays the details that you need to run a data load.

It includes the details of the source schema, the data entities that are loaded from the source schema, and the details of the target schema. You can choose the action that you want to apply on each data entity – recreate, truncate, append - and load tables in bulk to the target schema. When you run a data load, multiple jobs run in the background to complete the request.

You can run a data load in either of the following ways:

- Using the Data Load Detail Page.
- Running a Data Load from the Workflow Details Page.

> **Note:**
>
> Data load is not supported for Oracle Object Storage connections.

Running a Data Load from the Data Load Detail Page

The Data Load Detail page displays the information that you need to run a data load. You can apply different actions - incremental merge, incremental append, recreate, truncate, append - on the data entities before loading it to the target schema.

> **Note:**
>
> APPLIES TO: Data Transforms that is available as a separate listing on Marketplace called Data Integrator: Web Edition.
> If the data load is huge, you might want to increase the memory of the ODI Agent to avoid any issues. Follow the instructions in Increase the Memory of ODI Agent before you start to run the data load.

To run a data load from the Data Load Detail Page:

1. In the Data Load Detail page, select the data entities that you want to move to the target schema.
   To filter the list, you can do one of the following:
   - Enter the name or part of the name in the **Name** text box. This search returns data entities that include the specified string in the name. For example, the search string AD returns results such as ADMIN, ADDRESS, BADGE, UPGRADE, WORKLOAD, and so on.
   - Turn on the **Use Regular Expression** toggle to use pattern matching to search for particular strings of characters. For example, the search string CO.* returns results such as CONTACT, COUNT, COUNTRY and so on.

   You can select multiple data entities and load them to the target schema. You can also sort the displayed list using the following options:
   - All - Displays all the data entities that match the search criteria.
   - Selected - Displays all the rows that you selected.
   - Unselected - Displays all the unselected rows.
   - Invalid – Displays all the invalid rows.

   > **Note:**
   >
   > These options display the list of data entities based on the search criteria. To view the list of all data entities, clear any applied filters.

2. Click on the required icon to choose any of the following actions:
   - Incremental Merge - Updates the data in the selected column by comparing the source table with the target table based on the specified merge key. To use this option, select the column that you want to merge and then select the merge key. Click the Validate icon (✓) to validate the selected values.
   - Incremental Append - Updates data in the selected column in the target schema. To use this option, select the column that you want to update and click the Validate icon (✓) to validate the selection.
   - Recreate – If the table is already present in the target schema, drops the existing table and recreates it.

     > **Note:**
     >
     > This option is not available for data entities that are loaded using OCI GoldenGate.

   - Truncate – If the table is already present in the target schema, deletes all the data from the selected table. Nothing is dropped.

> **Note:**
>
> For Delta Share data loads the Data Load Detail page only includes the option. You cannot apply different actions - incremental merge, incremental append, recreate, truncate, append - on the data entities before loading it to the target schema. This is to make sure that the data is consistent between the Delta Sharing server and the target schema.

- Append – If the table is already present in the target schema, adds rows to the table.
- Do Not Load – Skips the selected data entity from the data load job. After you click **Save**, these data entities are no longer available for future data load jobs.

You can select multiple data entities and apply different actions. The unsaved rows are highlighted in bold.

> **Note:**
>
> These options are not available for Delta Share connections.

3. To specify how you want to store the source column names in the target tables, click **Advanced Options**, which is on the right side of the Data Load Detail page. Choose one of the following:
   - **Retain original names by enclosing all names with delimiters** - Creates column names with the same names as is from the source tables in the target table.
   - **Use no delimiters** - This is the default selection. Converts all the column names to upper case and replaces spaces and special characters with underscores.
     The following options are applicable to reserved words such as `Date`, `Timestamp`, `Start`, and so on.
     - **Enclose with delimiters** - This is the default selection. Encloses column names that are reserved words with delimiters (not all column names).
     - **Use a prefix** - Adds the specified prefix to column names that are reserved words (not all column names).

     For column names that have the same name after conversion, the names are suffixed with a numeric value to maintain uniqueness. For example column names

     `Date, date, DATE, Item_@Code, Item$$Code, Item%%Code`

     are created in the target table as

     `DATE, DATE_0, DATE_1, ITEM__CODE, ITEM__CODE_0, ITEM__CODE_1.`

> **Note:**
>
> Once the data load is run, the selected options are applied and retained for all subsequent runs. You cannot change the configuration.

4. Click to save the changes. A green checkmark ( ) in the row indicates that the changes are saved.
5. To start the data load,

- Click .
- For GoldenGate data loads, click .

A confirmation prompt appears when the data load starts successfully.

To check the status of the data load, see the Status panel on the right below the Target Schema details. For details about the Status panel, see Monitor Status of Data Loads, Data Flows, and Workflows. This panel shows links to the jobs that execute to run this data load. Click the link to monitor the progress on the Job Details page. For more information about jobs, see Create and Manage Jobs.

Running a Data Load from the Workflow Details Page

You can add multiple data loads to a workflow along with data flows or workflows and run them as separate steps. The left panel of the Workflow Details page lists the data flows, workflows, and data loads that are available for use.

For data loads, the left panel lists the following two folders:

- **Data Loads** - This folder lists all the data loads that you have created in the local Data Transforms instance. When you select a step in the workflow that is a local data load, the Properties Panel available on the right side of the design canvas displays the Type as `Data Load` and the Linked Object as Home>`<nameofDataLoad>`.
- **Remote Data Loads** - This folder lists all the data loads that you have created in a remote Data Transforms instance. See Create a Data Transforms Connection for Remote Data Load. When you select a step in the workflow that is a remote data load, the Properties Panel available on the right side of the design canvas displays the Type as `Remote Data Load` and the Linked Object as `Name of the Data Transforms connection` > `<nameofDataLoad>`.

After you have added the data loads to the workflow, click to execute them.

- Increase the Memory of ODI Agent

Increase the Memory of ODI Agent

APPLIES TO: Data Transforms that is available as a separate listing on Marketplace called Data Integrator: Web Edition.

If the data that you are loading from the source schema is huge, then you may want to increase the memory of the ODI Agent to avoid OutOfMemory exception errors.

To increase the memory of the ODI Agent:

1. Edit the `/u01/oracle/transforms_home/common/scripts/jettyServer.sh` file.
2. Add the `java -Xms1024m -Xmx4096m` parameter.
3. Restart the jetty server. Log in as OPC user and execute the following commands:

```
ssh -i <path to id_rsa> opc@<Instance IP>
sudo su
systemctl stop|start jettyserver.service
exit
```

Work with Data Entities

A Data Entity is a tabular representation of a data structure.

It includes Database Tables or Views that can be used in a mapping as a source or target. They are simply the metadata for sources and targets. They are used in creating data flows.

You can add Data Entities to your newly created project in one of the following two ways:

- Importing Data Entities
- Creating Data Entities

All the newly created or imported Data Entities along with their details are displayed in the **Data Entities** page. The details include:

- **Name** of the Data Entity
- **Connection** for which the Data Entity was created
- **Schema** to which the Data Entity is associated
- Click the **Actions** icon (⋮) next to the selected Data Entity to perform the following operations:
  - Select **Edit**, to edit the existing details.
  - Select **Preview**, to preview the selected Data Entity. If the data entity belongs to an Oracle database, you can also view statistics of the table. See View Statistics of Data Entities for more details.
  - Select **Delete**, to delete the selected Data Entity.
- To delete the Data Entities in bulk, in the Data Entities page, select the check boxes of the respective Data Entities and click **Delete**.
- You can also search for the required Data Entity to know its details based on the following filters:
  - **Name** of the Data Entity
  - **Connection** for which the Data Entity was created
  - **Schema** to which the Data Entity is associated
  - **Tag** that is associated with the Data Entity
- Import Data Entities
  The easiest and most common way to create a Data Entity is by importing its structure from the connection type (for example, Oracle database).
- Create Data Entities
  You can manually create or update the Data Entities through the Oracle Data Transforms interface.
- Create Data Entities within the Data Flow editor
  If you have already created or imported your target data entity, then you would drag the data entity onto the Design Canvas and complete the column mappings and options.
- View Statistics of Data Entities
  The Preview tab displays detailed statistics of each data entity.

Import Data Entities

The easiest and most common way to create a Data Entity is by importing its structure from the connection type (for example, Oracle database).

To import existing Data Entities:

1. From the left pane of the Home page, click the **Data Entities** tab.
   Data Entities page appears.
2. Click **Import Data Entities**, to import the existing data entities.
   Import Data Entities page slides-in.
3. Select the **Connection** followed by **Schema** and then finally select the **Type of Objects** you want to import. For Oracle Object Storage connections, the Schema drop-down lists the name of the bucket that you specified in the URL when you created the connection.
4. [For Oracle Business Intelligence Cloud Connector (BICC) connections only] From the **Offerings to import for collection**, choose the offerings whose data stores you want to import. You must select at least one offering to import the BICC public view objects (PVO).

   > **Note:**
   >
   > The import of BICC PVOs can take a long time depending on the number of selected objects. To improve performance, Oracle recommends that you use a mask to filter and limit the number of PVOs that you want to import.

5. Choose a **Mask**/filter if you don't want to import every object in the schema. Depending on the Connection Type, you will be presented with further options for importing.

   > **Note:**
   >
   > For Oracle Object Storage connections, this value is case-sensitive. If **Batch similar files** is set to `True`, all the files that match the mask and have the same structure are grouped together into a single data entity.

6. [For Oracle Financials Cloud connections only] From the list in the **Resources** section, select the items that you want to import. When the import process completes, a table is created for each selected resource.
7. [For REST server connections only] In the **Resources** section, do the following:
   - In the **Resource URI** field, enter the URL of the REST service you want to import resources from.
   - Click the + icon.
   - In the **Name** column enter an identifier for the resource.
   - In the **Operation URI** column enter the URI of the resource.
   - Click **Test Resource** to check whether the entries are valid.
8. Click **Start**.
   A Job is created and the corresponding Job ID is displayed for you to track the session. Click the Job ID to view the details of the job.

Upon successful execution of the job, all the selected Data Entities are imported. Click the Refresh icon present at the right corner of the Data Entities page, to see the new imported Data Entities.

Create Data Entities

You can manually create or update the Data Entities through the Oracle Data Transforms interface.

Data entities should possess the corresponding objects in the source connection to be used as a source in a data flow. Usually the import process makes sure that these objects are in coordination. However, whenever you manually create or update Data Entities always make sure to check if both the definitions are in coordination with each other.

When you use a Data Entity as a target then it doesn't have to exist previously in the target connection and can be created as a part of Data Flow execution.

To create a new Data Entity:

1. From the left pane of the Home page, click the **Data Entities** tab.
   Data Entities page appears.
2. Click **Create Data Entity**, to create a new data entity.
   Create Data Entity page appears.
3. In the Name text box, enter the name of the new Data Entity that you are creating.
4. From the **Connection** drop-down, select the required connection from which you wish to add the newly created Data Entity.

   > **Note:**
   >
   > Oracle Financials Cloud connections are not listed here because you cannot manually create data entities for such connections. You can only import data entities from Oracle Financials Cloud REST endpoints using the Import Data Entities page. See Import Data Entities.

5. In the **Schema** drop-down, all schema corresponding to the selected connection are listed in two groups.
   - New Database Schema (ones that you've not imported from before) and
   - Existing Database Schema (ones that you've imported from before and are potentially replacing data entities).

   From the **Schema** drop-down, select the required schema.

   > **Note:**
   >
   > For Oracle Object Storage connections, the Schema drop-down lists the name of the bucket that you specified in the URL when you created the connection.

6. From the **Type** drop-down, select the data entity type.
   - **Table**: To define the table structure for the newly created Data Entity, click the + icon to add columns. For each column, you can specify parameters such as Name, Data Type, Length, Scale, Not Null. Double click on the cell to configure the value.
     Click the 'x' icon, to delete a row. Click the Up and Down arrows to sort the table rows.

- **Inline View**: To create the data entity using inline code, enter the Select statement in the **Query** tab. For example, `SELECT * FROM CUSTOMER`. Click **Validate**.
  The **Columns** tab displays a read-only list of the columns that the query returns. Click the **Preview** tab to see the column data.

7. In the **Tags** text box, enter a tag of your choice. You can use tags to filter the Data Entities displayed in the Data Entity Page.
8. For Oracle Object Storage connections, this page displays the following options:
   - **Contents** – Select the CSV file that contains the data you want to import. The metadata displayed in the preview table, such as the data type and length of columns, is based on the first row of the CSV file. Make sure that the CSV file has a header line. The header should contain only alphanumeric characters and no special characters.
   - **Group Files** – Select this check box if you want to group data from multiple CSV files into one data entity. For example, say you want to merge data from Employee_Data1.csv, Employee_Data2.csv, and Employee_Data3.csv into a single data entity.
   - **Resource Name** – Use this option along with **Group Files**. Enter the value you want to use to identify the files. The resource name should be a regular expression. You can use only an asterisk (*) as a wildcard character in the resource name. For example, `Employee_Data*.csv`.
9. For Oracle database connections, you can mark the data entities as a feature group. Expand **Advanced Options** and click the **Treat as Feature Group** checkbox.
10. Click **Save**.
    The new Data Entity is created.

Create Data Entities within the Data Flow editor

If you have already created or imported your target data entity, then you would drag the data entity onto the Design Canvas and complete the column mappings and options.

To create the definition of a new target table while in the Data Flow editor,

1. Select the component at the end of your data flow.
2. Click the Add Data Entity icon  present on the top right corner of the target component.
3. **Add Data Entity** page appears allowing you to configure the following details of the target component:
   General tab
   - In the **Name** text box, enter the name of the newly created Data Entity.
   - The **Alias** text box is auto-populated with the name of the newly created Data Entity.
   - From the **Connection Type** drop-down, select the required connection from which you wish to add the newly created Data Entity.
   - It loads the server name coined at the time of connection creation. From the **Server** drop-down, select the required server name from which you wish to add the newly created Data Entity.
   - From the **Schema** drop-down, select the required schema.
   - Click **Next**.

   Columns tab

   It allows you to create, remove or edit the column definitions.

- Click the  Add Columns icon, to add new columns to the newly created Data Entity.
  A new column is added to the displayed table.
- The table displays the following columns:
  - Name
  - Data Type - Click the cell to configure the required Data Type.
  - Scale
  - Length
  - Actions - Click the cross icon to delete the created column.
- To delete the columns in bulk, select the columns and click the  Delete icon.
- To search for the required column details, in the Search text box enter the required column name and click enter. The details of the required column are displayed.
- Click **Next**.

Preview Data Entity tab

It displays a preview of all the created columns and their configured details. If the data entity belongs to an Oracle database, you can also view statistics of the table. See View Statistics of Data Entities for more information.

4. Click **Save**.
   The new target Data Entity is created.
5. Expand the **Properties Panel** in the right pane to view the following settings of the created components:
   - General - Displays the Name of the component along with its Connection and Schema details.
   - Attributes - Displays the details of all the attributes associated to the component.
   - Column Mapping - Click Auto Map to map all the columns automatically.
   - Preview - Click to have a preview of the component.
   - Options - Change the options as appropriate.

View Statistics of Data Entities

The Preview tab displays detailed statistics of each data entity.

> **Note:**
>
> This feature is available for Oracle database tables only.

You can view the statistics of a selected data entity in one of following ways:

- In the Data Entities list, click the **Actions** icon (⋮) next to the Data Entity and click **Preview**. Select the **Statistics** tab to view the statistics of the selected data entity.

- On any data flow click on any source or target data entity, and expand the properties panel in the right pane. Click **Preview**.

The statistical data is presented as follows:

- The total number of rows and columns in the data entity is displayed at the top.
- The statistics panel displays the thumbnail graphs for each column with information about the Min, Max, Distinct, and Null values.
- Two types of thumbnail representations are displayed based on the histogram:
  - A bar chart represents data for frequency and top-frequency histograms. The bar chart show the first top 10 values for the number of rows in the table.
  - A table lists data for Hybrid and Height-Balanced histograms. The table displays the entire data and is scrollable. The table displays the range for the values and the percentage of rows in each range.
- You can click each thumbnail to view the statistics of the column in a new browser tab.
- The detailed view of each chart also shows the type of histogram.

Work with Projects

A project is the top-level container, which can include multiple folders to organize your data flows or work flows into logical groups.

The Projects page displays all the existing projects, which includes projects created by all users not just the logged in user.

You can perform the following operations of a Project folder:

- View Details
- Edit the name
- Delete

Creating a Project

To create a new project:

1. Click **Create Project** on the Projects page.
2. On the **Create Project** page, provide a project name.
3. Click **Create**.
   After creating the project, you're redirected to the **Projects** page.

Viewing Project Details

To view project details:

1. On the Projects page, select a project and click **View Details** from the project's **Actions** icon (⋮) or select a project to open its details page.
2. It displays all the resources associated to the selected project. It includes details such as **Data Flow Name** and **Folder** details for the selected project.
3. Click the **Actions** icon (⋮) of the respective data flow to edit, rename, copy, change folder, start or delete the required data flow of the selected project.

Deleting a Project

When you delete a project, you also delete the resources it contains. Once you delete a project, it cannot be restored.

> **Note:**
>
> Be sure to review all contained resources before you delete the project.

To delete a project, on the Projects page, select Delete from the **Actions** icon (⋮) for the project you want to delete or select a project to open its details page, and then click **Delete**.

In the **Delete Project** dialog, click **Delete** to confirm the delete operation.

Create a Data Flow

A data flow defines how the data is moved and transformed between different systems.

A data flow in Data Transforms connects sources to targets through a flow of components such as Join, Filter, Aggregate, Set, Split, and so on. See Supported Database Functions for more information.
When you run a data flow, Data Transforms uses the joins, filters, mappings, and constraints to transform source data and load it to target tables. Note that you can run only one execution flow at a time. You cannot put multiple flows on a Data Flow and a flow cannot diverge into multiple flows.

You can create data flows in any of the following ways:

- From the Projects page
- From the Data Flows page within a project
- From the Home page

From the Projects page

To create a data flow from the Projects page,

1. On the Projects page, click **Create Data Flow**.
   Create Data Flow page appears:
2. In the **Name** field, enter a name for the new data flow.
3. Select **Create New Project**, if you wish to create new project folder for the newly created data flow.
4. Else, click **Add to Existing Projects**, if you wish to add the newly created data flow to an existing project folder.
5. If you have selected **Create New Project** for the previous option, in the **Project Name** field, enter the name of the newly created project.
6. Else, if you have selected **Add to Existing Projects** for the previous option, select the required project from the **Project Name** drop-down arrow.
7. In the **Description** field, enter a description for the newly created data flow.
8. Click **Create**.

From the Data Flows page within a project

To create a data flow from the Data Flows page within a project,

1. On the Projects page click the project tile you wish to create a new data flow for. The **Project Details** page appears.
2. In the **Data Flows** page, click **Create Data Flow**.
3. Provide the **Name** and **Description** of the new data flow.
4. Click **Next**.
5. To define your source connection, from the **Connection** drop-down, select the required connection from which you wish to add the data entities.
6. In the Schema drop-down, all schema corresponding to the selected connection are listed in two groups:
   - Existing Schema (ones that you've imported into Oracle Data Transforms) and
   - New Database Schema (ones that you've not yet imported).

   Select the schema that you want to use from the drop down. For Oracle Object Storage connections, the Schema drop-down lists the name of the bucket that you specified in the URL when you created the connection.
7. Click Save.
   The Data Flow Editor appears that allows you to create a new data flow.

From the Home page

To create a data flow from the Home page,

1. On the Home page, click **Transform Data**. The Create Data Flow page appears.
2. Provide the **Name** and **Description** of the new data flow.
3. Select a project name from the drop-down. Alternatively, click the + icon to create a project.
4. Click **Next**.
5. From the **Connection** drop-down, select the required connection from which you wish to add the data entities. Alternatively, click the + icon to create a new connection.
6. In the Schema drop-down, all schema corresponding to the selected connection are listed in two groups:
   - Existing Schema (ones that you've imported into Oracle Data Transforms) and
   - New Database Schema (ones that you've not yet imported).

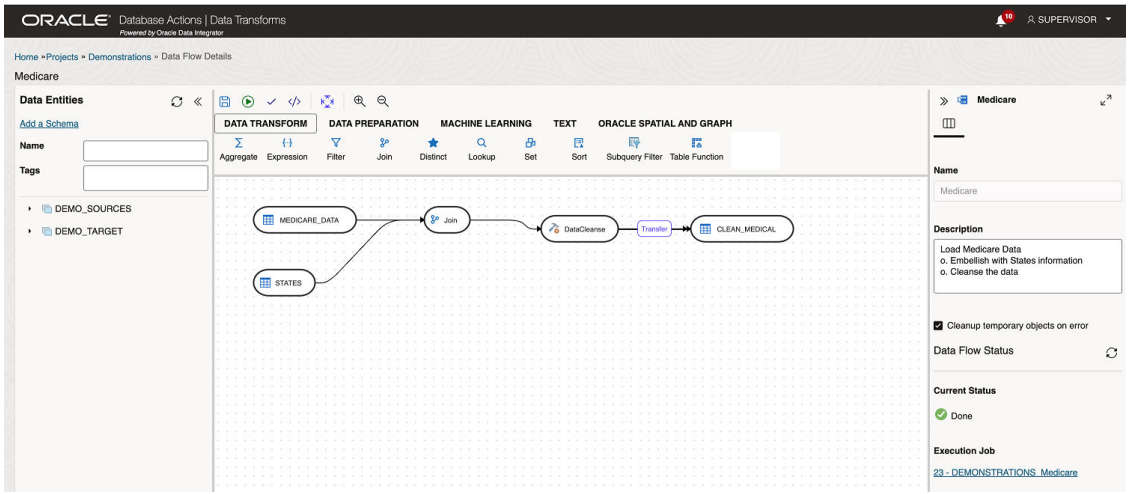
   Select the schema that you want to use from the drop down.
7. Click **Save**.

- About Data Flow Editor
  The Data flow editor is divided into five parts, the Data Entity Panel, the Transformations Toolbar, the Design Canvas, the Properties Panel, and the Status Panel.
- Add Components
  Add the data entities and database functions to the Design Canvas, and connect them in a logical order to complete your data flows.

- Map Data Columns
  When you connect the source data entity with the target data entity, the column names are automatically mapped by the column names. You have a choice to map the columns by Position or by Name or map the columns manually using the Expression Editor.
- Validate and Execute a Data Flow
  After your mappings are ready, you can proceed to validate and execute the data flow.

About Data Flow Editor

The Data flow editor is divided into five parts, the Data Entity Panel, the Transformations Toolbar, the Design Canvas, the Properties Panel, and the Status Panel.



- **Data Entities Panel**: The data entity panel displays the Data Entities that are available to use in your Data flows. The displayed list can be filtered using the Name and Tags fields. The panel includes options that let you add schemas, import data entities, remove any of the schemas that are associated with the data flow, and refresh data entities. See Add Components for information about how to use these options.
- **Database Functions Toolbar**: The Database Functions toolbar display the database functions that can be used in your data flows. Just like Data Entities, you can drag and drop the Database tools you want to use on the design canvas. See Supported Database Functions for more information.
- **Design Canvas**: The design canvas is where you build your transformation logic. After adding the Data Entities and Database Functions to the design canvas, you can connect them in a logical order to complete your data flows.
- **Properties Panel**: The properties panel displays the properties of the selected object on the design canvas. The Properties Panel is grouped into four Tabs. **General**, **Attributes**, **Preview Data**, **Column Mapping**, and **Options**. Not all tabs are available as they vary based on the selected object. See Component Properties to know more about these options.
- **Status Panel**: When you run a data flow, the Status Panel shows the status of the job that is running in the background to complete the request. You can see the status of the job that is currently running or the status of the last job. For more information about the Status panel, see Monitor Status of Data Loads, Data Flows, and Workflows.

After designing the required data flow,

- Click  , to save the created/designed data flow.
- Click  , to align the nodes of the designed data flow.
- Click  , to execute the created data flow.
- Click  , to validate the created data flow.
- Click   , to maximize or minimize the created data flow diagram in the design canvas.
- Supported Database Functions
  Oracle Data Transforms supports various database functions that you can drag and drop on the Design Canvas to connect components within a data flow.

Supported Database Functions

Oracle Data Transforms supports various database functions that you can drag and drop on the Design Canvas to connect components within a data flow.

The Database Functions toolbar in the Data Flow editor includes the following database functions that can be used in your data flows:

1. **Data Transformation**
   It contains the following components:
   - Aggregate
   - Expression
   - Filter
   - Join
   - Distinct
   - Lookup
   - Set
   - Sort
   - Subquery Filter
   - Table Function
2. **Data Preparation**
   It contains the following components:
   - Data Cleanse
   - Substitution
   - Equi_Width Binning
   - Quantile Binning
   - Lead
   - Lag
   - Replace
3. **Machine Learning**
   It contains the following components:
   - Prediction

- Outlier Detection

4. **Text**
   It contains the following components:
   - REGEXP COUNT
   - REGEXP INSTR
   - REGEXP SUBSTR
   - REGEXP REPLACE
   - Edit Distance Similarity
   - Contains
5. **Oracle Spatial and Graph**
   It contains the following components:
   - Buffer Dim
   - Buffer Tol
   - Distance Dim
   - Distance Tol
   - Nearest
   - Simplify
   - Point
   - Geocode Tools:

     > **Note:**
     >
     > The following Geocode Tools work only in non-Autonomous Database environment.

     - Geocode As Geometry
     - Geocode
     - Geocode Addr
     - Geocode All
     - Geocode Addr All
     - Reverse Geocode

     > **Note:**
     >
     > The following Geocode Tool works only in an Autonomous Database environment.

     - Geocode Cloud
   - Spatial Join

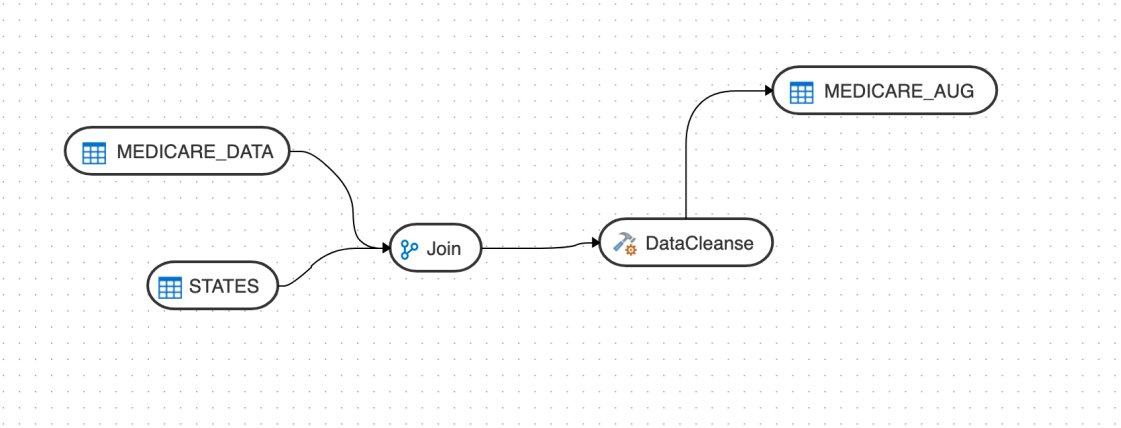
Add Components

Add the data entities and database functions to the Design Canvas, and connect them in a logical order to complete your data flows.

To add components to your data flow:

1. In the Data Entities panel, click **Add a Schema** to add schemas that contain the data entities that you want to use in the data flow.
2. In the Add a Schema page, select the connection and schema name.
3. Click **Import**.
4. In the Import Data Entities page, select the Type of Objects you want to import. Choose a Mask/filter if you don't want to import every object in the schema and click **Start**.
5. The Data Entities panel lists the imported data entities. The panel includes various options that let you do the following:
   - Refresh Data Entities – Click the Refresh icon to refresh the displayed list.
   - **Name** - Search for data entities by name.
   - **Tags** - Filter the data entities by the name of the tag used.
   - **Import Data Entities** - Right-click the schema to see this option. Use this option to import the data entities.
   - **Remove Schema** - Right-click the data entity to see this option. Use this option to remove the schema from the list. Note that this option does not delete the schema, it only removes the association of the schema with this data flow.
6. Similarly add more schemas to the Data Flow, if required.
7. Drag the required Data Entities that you want to use in the data flow and drop them on the design canvas.
8. From the **Transformations** toolbar, drag the transformation component that you want to use in the data flow and drop them on the design canvas.
9. Select an object on the design canvas, and drag the **Connector** icon ( ) next to it to connect the components.
10. After saving the data flow, there may be a Transfer icon overlaid on one or more of the component connections. This indicates that ODI has detected an additional step and it is required to move the data between data servers. You can click on this Icon to view properties associated with this step.

For example:



- Component Properties
  The Properties Panel displays various settings for components selected in the Design Canvas.

Component Properties

The Properties Panel displays various settings for components selected in the Design Canvas.

Depending on the component selected, you may see any of the following icons:

- General ( ) - Displays the name of the component along with its connection and schema details. You can edit some of these properties.
- Attributes ( ) - Displays the details of all the attributes associated with the component.
- Column Mapping ( ) - Allows you to map all the columns automatically. See Map Data Columns for more information.
- Preview ( ) - Displays a preview of the component. For Oracle tables, you can also view the statistics of the selected data entity. See View Statistics of Data Entities for details about the statistical information available.
- Options ( ) - Displays options such as
  - Truncate Table - Replaces any existing target table content with new data.
  - Append - Inserts records from the flow into the target. Existing records are not updated.
  - Incremental - Integrates data in the target table by comparing the records of the flow with existing records and updating the records when their associated data is not the same. Those that don't yet exist in the target are inserted.
    The option includes an Auto compression feature that is set to `True` by default. For data flow jobs that use the Incremental Update mode to load data onto a compressed Oracle target partition, the Auto compression feature recompresses the modified target partitions after the load completes successfully. For table partitions that are not originally compressed, the compression is skipped irrespective of whether Auto compression is set to true.

> **Note:**
>
> The Auto compression option is available to the ADMIN user or to a user with the DWROLE role. For data flows that have schema users other than ADMIN you need to either assign the DWROLE to the user or disable Auto compression to avoid execution errors.

Map Data Columns

When you connect the source data entity with the target data entity, the column names are automatically mapped by the column names. You have a choice to map the columns by Position or by Name or map the columns manually using the Expression Editor.

To map columns by Position or by Name:

1. Select the target Data Entity.
2. Click the arrow icon present on the top right corner to expand the Properties Panel. This will give you more space to work with.
3. In the **Properties Panel**, click the **Column Mapping** icon ( ).
4. To map the columns by Position or by Name, from the **Auto Map** drop-down menu, select **By Position** or **By Name**.

To map the columns manually:

1. From the **Auto Map** drop-down menu, select **Clear** to clear the existing mappings.
2. Drag and drop the attributes from the tree on the left to map with the Expression column.
3. To edit an expression, click the **Edit** icon of the respective column. The **Expression Editor** appears allowing you to perform the required changes (for example, you can just add an expression-"UPPER" or open the Expression Editor to edit the expression).

> **Note:**
>
> Use the expression editor only if you have complex expressions for a particular column.

4. Click **OK**.

Validate and Execute a Data Flow

After your mappings are ready, you can proceed to validate and execute the data flow.

Do the following:

1. Click Save.
   After saving, if data needs to be staged before transforming, Transfer button is added to one or more links. You can click these buttons to set more options, if available.
2. Click the **Simulate Code** icon ( ) if you want to check the code that will run to complete the tasks that are performed when you execute the data flow job. The source and target details are displayed in different colors for ease of reference. This is handy if you want to check if the mapping is correct before you run the job or if the job fails. Note that the code

cannot be used for debugging. For detailed information about the job, see the Job Details page.

3. Click the **Validate** icon ( ) in the toolbar above the design canvas to validate the data flow.

4. After a successful validation, click the **Execute** icon ( ) next to the **Validate** icon to execute the data flow.
A message appears that displays the execution Job ID and name. To check the status of the data flow, see the Status panel on the right below the Properties Panel. For details about the Status panel, see Monitor Status of Data Loads, Data Flows, and Workflows. This panel also shows the link to the Job ID that you can click to monitor the progress on the Jobs page. For more information, see Create and Manage Jobs.

   For data flows created using Oracle Object Storage connections, the data from the source CSV file is loaded into the target Oracle Autonomous Database. You can also export data from an Oracle Autonomous Database table to a CSV file in Oracle Object Storage.

Schedule Data Flows or Workflows

You can schedule workflows and data flows to run at specified time interval.

Each schedule allows you to specify a start date and time, and execution frequency.

To schedule a data flow or a workflow:

1. In the left pane, click **Schedules**.
2. Click **Create Schedule**.
3. From the **Resource** drop-down menu, select **Data Flow** or **Workflow** as appropriate.
4. From the **Resource Name** drop-down list, select a data flow or workflow that you want to schedule.
5. From the **Frequency** drop-down list, select the time frame in which you wish to execute the created schedule. You can schedule it On Startup, Simple, Daily, Hourly, Weekly, Monthly or Yearly. Based on the selected frequency, **Time** field appears allowing you to schedule the selected Data Flow or Workflow. For detailed information refer to the below table:

Frequency	Time Values
On Startup	NIL
Simple	Click Select Date Time icon next to **Date and Time** field, to select the required date and time in `MM/DD/YY` and `HH:mm` format.
Daily	Click the clock icon next to **Time** field, to select the required time in `HH:mm` format.
Hourly	Click up and down arrows next to **Time** field, to select the required time in `Minutes:Seconds` format.
Weekly	• **Run Every** parameter appears with check-boxes for all the days of the week. • Click the clock icon next to **Time** field, to select the required time in `HH:mm` format.

Frequency	Time Values
Monthly (day of the month)	• Select the required date from the **Monthly Date** drop-down box. • Click the clock icon next to **Time** field, to select the required time in `HH:mm` format.
Monthly (week day)	• From the **Monthly date** drop down box select the required value. • From the **Week day** drop-down box, select the required weekday in which you wish to schedule the job.
Yearly	• From the **Month** drop down box, select the required month in which you wish to schedule the job. • From the **Monthly date** drop down box, select the required date in which you wish to schedule the job. • Click the clock icon next to **Time** field, to select the required time in `HH:mm` format.

6. In the **Number of Attempts on Failure** text box, enter the required value or click the up or down arrows next to the text box, to select the required value. This value denotes the number of retry attempts that should happen after the schedule failure.
7. **Stop Execution After** field denotes the time after which the schedule has to stop after executing it. It can be in **Hours**, **Minutes** or **Seconds**. Select the value and its unit.
8. Select the status of the scheduled Data Flow or Workflow from the **Status** options. It can be **Active**, **Inactive** or **Active for Period**. Through **Active for Period** option you can configure the exact time frame of the created schedule. It's Starting and Ending Date with time, time interval in which you wish to run the schedule and exceptions, if any.
9. After configuring the above details, click **Save**.
   The created schedules get listed in the **Schedules** page along with the following details:
   - Resource
   - Status
   - Frequency
   - Valid or Invalid

   Click the Actions menu (⋮) of the respective schedule to perform the following operations:
   - Click **Edit** to edit the details of the created schedule.
   - Click **Disable** to disable the created schedule. The status of a schedule becomes inactive when you disable it and gets displayed in the Scheduled list. Select the schedule and click **Enable**, to enable it whenever required. You can enable it again for a specific period, through **Enable for Period** option.
   - Click **Validate** to validate the created schedule.
   - Click **Delete** to delete the created schedule. Upon confirmation the created schedule is deleted.

To schedule a data flow or a workflow specific to a project:

- Click the Project title displayed on the **Projects** page.
  Project Details page appears.

- In the left pane, click **Schedules**.
  Schedules page appears. It displays all the schedules pertaining to the selected project.

  > **Note:**
  >
  > This page is project specific and displays only the Data Flows and Workflows pertaining to the selected project.

- To create a new schedule, click **Create Schedule**.
- Enter all the required details and click **Save**.
  The new schedule is created and is added to the existing list of schedules for the project.

Monitor Status of Data Loads, Data Flows, and Workflows

When you run a data load, data flow, or workflow Oracle Data Transforms runs jobs in the background to complete the request. You can view the status of the job in the panel on the bottom right of the Data Load Details, the Data Flow Editor, and the Workflow Editor page.

This panel includes the following:

- A Refresh icon ( ) to refresh the displayed status.
- The current status. You can see any of the following statuses:
  - Not Started – If no data load, data flow, or workflow has been executed yet.
  - Running – When the job starts.
  - Done – When the job completes.
    The status of the last job run persists in this panel till you run a new data load, data flow, or workflow.

    > **Note:**
    >
    > For data loads created using OCI GoldenGate, the status panel shows the link to the GoldenGate Deployment Console and the status of the Extract and Replicat processes.

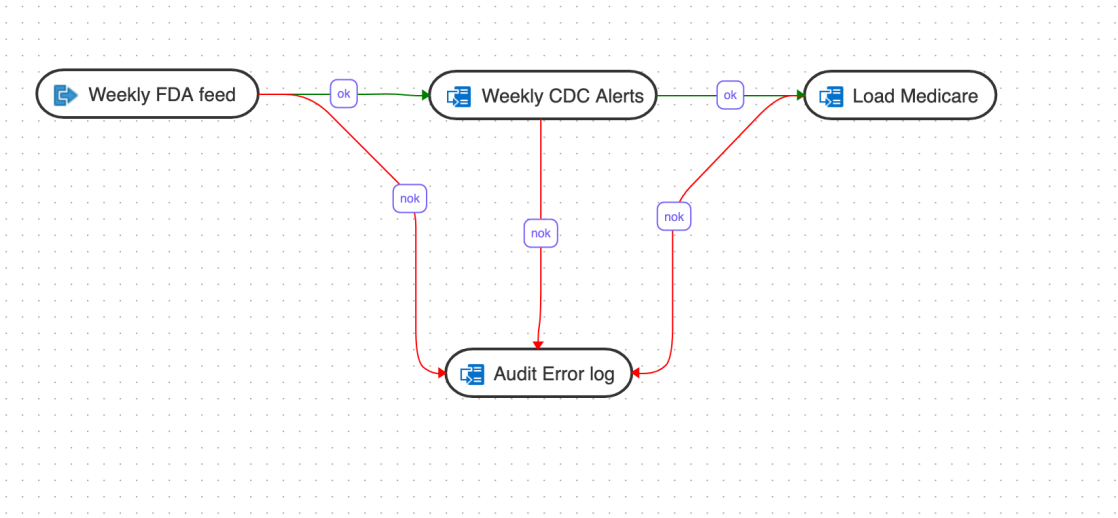
- The name of the job and a link that takes you to the Jobs Details page where you can see detailed information about the job and monitor the execution status. For more information, see Create and Manage Jobs.
  When you run a data load, multiple jobs run in the background to complete the request. This panel shows a link for each job that executes when running a data load.

Introduction to Workflows

A workflow is made up of multiple flows organized in a sequence in which they must be executed.

Each data flow is executed as a step. You can also add workflows as well as SQL queries as steps within a workflow. When you execute a workflow, a data flow either succeeds or fails. Depending on whether the first data flow succeeds or fails, you can choose the next data flow that must be executed.

Here is an example of a workflow:



In this example, the workflow performs the following actions:

1. Execute the "Weekly FDA feed" data flow.
2. If the "Weekly FDA feed" data flow execution is successful, execute the "Weekly CDC alerts" data flow.
3. If the "Weekly CDC alerts" data flow execution is successful, execute the "Load Medicare" data flow.
4. If any of the above data flow fails, then execute the "Audit_error_log" data flow.

- Create a New Workflow
  You can add data flows, workflows, or data loads in a workflow.

Create a New Workflow

You can add data flows, workflows, or data loads in a workflow.

To create a new workflow for your project:

1. On the **Home** page, click the required Project title.
   You are navigated to the **Project Details** page.
2. In the left pane, click **Workflows**.
   The Workflow page appears.
3. On the **Workflow** page, click **Create Workflow**.
   The **Create Workflow** page appears.
4. In the **Name** field provide a name for the new workflow and click **Create**.
   The new workflow is created and listed in the **Workflow** page with its **Name** and **Folder** details.
5. Click the Workflow to configure the **Workflow Details**.
6. From the left panel drag the data flows, workflows, or data loads that you want to run in the workflow. If you have connected to any other Data Transforms instance, you can also add data loads that you have created in that Data Transforms instance. See Create a Data Transforms Connection for Remote Data Load and Run a Data Load for more information.
7. Select either the ok (green arrow) icon, the not ok (red arrow) or the ok/not ok (black arrow) in the toolbar.
   This defines the mode that subsequent links drawn on the canvas will be set to.
   - ok (green) defines the success path.

- not ok (red) defines the failure path.
- ok/not ok (black) defines a path that is followed on success or failure.

Use the Sleep (clock icon) to add a delay in the workflow. Drag the Sleep icon on the canvas and connect it in the flow with either the ok (green), not ok (red) or ok/not ok (black) links. This will add a delay at that point in the flow.

8. If you wan to add a SQL or PL/SQL query as a step in the workflow, do the following:

   - Drag the SQL icon () on the canvas.
   - Double click the SQL step in the editor to open the step properties page.
   - Select the **Attributes** tab.
   - From the **Connection** drop-down select the connection you want to run the query on.

     > **Note:**
     >
     > The drop-down lists only Oracle database connections.

   - In the **SQL** textbox add the query that you want to run.

9. Select the step and click the **Connector** icon () next to it to connect it with the next step.
10. After defining all the required Workflow details,

    - Click  , to save the created/designed workflow.
    - Click  , to align the nodes of the designed workflow.
    - Click  , to execute the created workflow.
    - Select a single step in the canvas and click the **Execute Step** icon (), to execute only the selected data flow or workflow.
      To check the status of the workflow, see the Status panel on the right below the Properties Panel. For details about the Status panel, see Monitor Status of Data Loads, Data Flows, and Workflows. This panel shows the link to the Job ID that you can click to monitor the execution status on the Jobs page. For more information about jobs see Create and Manage Jobs.
    - Click   , to maximize or minimize the created workflow diagram in the design canvas.
      The newly created workflows get listed in the Project Details page. The following details are displayed:
      - Name of the Workflow
      - Folder corresponding to the workflow - Default Folder

      To view the details of a workflow, click the name of the workflow and you are navigated to the Workflow Details page.

In the left pane, you can search for the required Data Flow or Workflow using the Name filter. In the **Name** text box, enter the name of the required Data Flow or Workflow.

Select a step in the canvas and check the **Properties Panel** available at the right side of the design canvas to know the following properties of the selected step in a created Data Flow or Workflow:

- Name
- Linked Object

> **Note:**
>
> You cannot edit this field.

- Step -
  - First Step - Select the **First Step** check-box, to execute the selected step as the first step for execution in a Data Flow or Workflow.

> **Note:**
>
> You can select only a single step as a first step for execution in a Data Flow or Workflow.

- Number of attempts on Failure
- Time between attempts in seconds(s)
- Log steps in journal - Select any of the following options from the drop-down box:
  - Always
  - Never
  - Error

Create and Manage Jobs

When you execute a data load, data flow or workflow Oracle Data Transforms creates a job to complete the process in the background.

A job is made up of multiple steps that corresponds to an execution task.

The **Jobs** page by default, lists the execution job sessions that are running and completed for the present day as `Date` parameter is set to `Today` by default. You can click on any of the instance id from the list to see its details.

Create a Job from the Home page

To create a new job,

- On the Home page, click **Jobs** on the left pane to access the Jobs page.
- Click **Create Job**.
  The **Create Job** page slides-in.
- From the **Resource Type** drop-down, select the type of resource for which you wish to create a new job. It can be - Data Flow, Workflow or Schema.

- All the resources associated with the selected Resource Type get listed in the **Resource** field. From the **Resource** drop-down, select the required resource for which you wish to create a new job.
- Click **Create**.

A new job is created and is added to the existing list of jobs in the Jobs page.

For each session, you can see the Job Session **ID**, **Name**, **Status**, **Start Time** and **End Time** for each of the jobs that are executed.

Create a Job from the Projects page

To view the job sessions pertaining to a specific project:

- Click the Project tile displayed in the Projects page.
  Project Details page appears.
- In the left pane, click **Jobs**.
  Jobs page appears. It displays all the jobs and their details pertaining to the selected project.

> **Note:**
>
> This page is project specific and displays only the jobs pertaining to the selected project.

- To create a new job, click **Create Job**.
- Enter all the required details and click **Create**.
  A new job is created and is added to the existing list of jobs for the project.

View and Delete a Job

Click the Actions menu (⋮) to **View Details** of the jobs, **Delete** and **Rerun** the jobs, if required. If any of the steps failed, you can click on them to see the error details.

To delete a job, select the check box of the respective job session and click **Delete**. Upon confirmation the selected job gets deleted.

Search for a Job

You can also search for the required job session to know its details based on filters such as:

- **Name** - Name of the Job
- **Status** - Select the required session status from the list - All, Done, Error, Queued, Running, Waiting and Warning.
- **Date** - Select the date in which the required job session was executed - All, Today, Yesterday, Last Week, Last Month and Custom Range, which allows you to select specific From and To dates.

View Data Pump Jobs and Import Data with Data Pump

Use the Data Pump page to view Data Pump jobs and use the wizard to quickly create and run Data Pump import jobs.

See The Data Pump Page for more information.

Link Data

You can link to data in remote databases, in cloud storage, or in connected file systems or the local file system. You can also use Database Links to access objects on another database.

When you link to data in a remote database or in cloud storage or in connected file systems or the local file system, the target object produced is an external table or a view. When you select that target table or view on the Data Load, explore page in Database Actions, the source preview for the object shows the current data in the source object.

Linking to columns in a remote database, to files in cloud storage, or in connected file systems or the local file system can be useful when the source data is being continually updated. For example, if you have a database table with columns that are updated for each new sales transaction, and you have run a data load job that links columns of the source table to targets in your Oracle Autonomous Database, then those targets have the new sales data as it is updated in the source.

In addition, you can use Database Links to access objects on another database. Using Database Links, the other database need not be an Oracle Database system.

- Query External Data with Autonomous Database
  Describes packages and tools to query and validate data with Autonomous Database.
- Use Database Links with Autonomous Database
  You can create database links from an Autonomous Database to another Autonomous Database, to other Oracle databases, or to non-Oracle databases. In addition, you can create database links from other databases to an Autonomous Database.

Query External Data with Autonomous Database

Describes packages and tools to query and validate data with Autonomous Database.

External data is not managed by the database; however, you can use `DBMS_CLOUD` procedures to query your external data. Although queries on external data will not be as fast as queries on database tables, you can use this approach to quickly start running queries on your external source files and external data. Depending on the type of external table, you can validate external data using the `DBMS_CLOUD` validation procedures. The data validation procedures let you validate the source files for an external table so that you can identify problems and either correct the data in the external table or exclude invalid data before you use the data.

> **Note:**
>
> If you are not using `ADMIN` user, ensure the user has the necessary privileges for the operations the user needs to perform. See Manage User Privileges on Autonomous Database - Connecting with a Client Tool for more information.

- Query External Data
- Query External Data with ORC, Parquet, or Avro Source Files
  Autonomous Database makes it easy to access ORC, Parquet, or Avro data stored in object store using external tables. ORC, Parquet, and Avro sources have metadata embedded in them and the `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` procedure can utilize this metadata to simplify the creation of external tables.

- Query External Tables with Partitioning Specified in Source Files
  If you want to query multiple data files in the Object Store as a single external table and the files can be represented as multiple logical partitions, it is highly recommended to use an external partitioned table. Using an external partitioned table preserves the logical partitioning of your data files for query access.
- Query External Partitioned Data (with Partitioning Clause)
  If you want to query multiple data files in the Object Store as a single external table and the files can be represented as multiple logical partitions, then it is highly recommended to use an external partitioned table. Using an external partitioned table preserves the logical partitioning of your data files for query access. Use the procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` to create an external partitioned table.
- Query Hybrid Partitioned Data
  If you want to query internal data and multiple data files in the Object Store as single logical table you can use a hybrid partitioned table to represent the data as single object. Use the procedure `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` to create a hybrid partitioned table.
- Query External Data Pump Dump Files
  You can also query Oracle Data Pump dump files in the Cloud by creating an external table using `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`.
- Query Big Data Service Hadoop (HDFS) Data from Autonomous Database
  You can create database links to Oracle Big Data Service from Autonomous Database.
- Query External Data with Data Catalog
  Oracle Cloud Infrastructure Data Catalog is the metadata management service for Oracle Cloud that helps you discover data and support data governance. It provides an inventory of assets, a business glossary, and a common metastore for data lakes.
- Query External Data with AWS Glue Data Catalog
  Autonomous Database supports a system for synchronizing with an Amazon AWS Glue Data Catalog instance.
- Query Apache Iceberg Tables
  Autonomous Database supports querying Apache Iceberg tables.
- Validate External Data
  To validate any external table, you can use the procedure `DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE`.
- Validate External Partitioned Data
  To validate an external partitioned table, you can use the procedure `DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE`. This procedure includes a parameter that lets you specify a specific partition to validate.
- Validate Hybrid Partitioned Data
  To validate a hybrid partitioned table, you can use the procedure `DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE`. This procedure includes a parameter that lets you specify a specific partition to validate.
- View Logs for Data Validation
  To validate an external table, use the procedures `DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE`, `DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE`, and `DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE`.

Query External Data

To query data in files in the Cloud, you need to first store your object storage credentials in your Autonomous Database, and then create an external table using the PL/SQL procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`.

You can also use the procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` to query external data in attached file systems or in the local file system.

The procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` supports external files in the supported cloud object storage services, including:

- Oracle Cloud Infrastructure Object Storage
- Azure Blob Storage
- Amazon S3
- Amazon S3-Compatible, including: Oracle Cloud Infrastructure Object Storage, Google Cloud Storage, and Wasabi Hot Cloud Storage.
- GitHub Repository

The source file in this example, `channels.txt`, has the following data:

```
S,Direct Sales,Direct
T,Tele Sales,Direct
C,Catalog,Indirect
I,Internet,Indirect
P,Partners,Others
```

1. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DEF_CRED_NAME',
       username => 'adb_user@example.com',
       password => 'password' );
   END;
   /
   ```

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for creating external tables.

   See CREATE_CREDENTIAL Procedure for information about the `username` and `password` parameters for different object storage services.

2. Create an external table on top of your source files using the procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`.

   The procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` supports external files in the supported cloud object storage services. The credential is a table level property; therefore, the external files must be on the same object store.

For example:

```
BEGIN
   DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
    table_name =>'CHANNELS_EXT',
    credential_name =>'DEF_CRED_NAME',
    file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/channels.txt',
    format => json_object('delimiter' value ','),
    column_list => 'CHANNEL_ID VARCHAR2(2), CHANNEL_DESC VARCHAR2(20),
CHANNEL_CLASS VARCHAR2(20)' );
END;
/
```

The parameters are:

- `table_name`: is the external table name.
- `credential_name`: is the name of the credential created in the previous step. The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
- `file_uri_list`: is a comma delimited list of the source files you want to query.
- `format`: defines the options you can specify to describe the format of the source file.

  If the data in your source files is encrypted, decrypt the data by specifying the `format` parameter with the `encryption` option. See Decrypt Data While Importing from Object Storage for more information on decrypting data.
- `column_list`: is a comma delimited list of the column definitions in the source files.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

> **Note:**
>
> Autonomous Database supports a variety of source file formats, including compressed data formats. See DBMS_CLOUD Package Format Options and the `DBMS_CLOUD compression` format option to see the supported compression types.

You can now run queries on the external table you created in the previous step. For example:

```
SELECT count(*) FROM channels_ext;
```

By default the database expects all rows in the external data file to be valid and match both the target data type definitions as well as the format definition of the files. If there are any rows in the source files that do not match the format options you specified, the query reports an error. You can use `DBMS_CLOUD` parameters, like *rejectlimit,* to suppress these errors. As an alternative, you can also validate the external table you created to see the error messages and the rejected rows so that you can change your format options accordingly. See Validate External Data for more information.

For detailed information about the parameters, see CREATE_EXTERNAL_TABLE Procedure.

See DBMS_CLOUD URI Formats for more information on the supported cloud object storage services.

- External Table Metadata Columns
  The external table metadata helps you determine where data is coming from when you perform a query.

External Table Metadata Columns

The external table metadata helps you determine where data is coming from when you perform a query.

The external tables you create with `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`, `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`, or `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from.

- `file$path`: Specifies the file path text up to the beginning of the object name.
- `file$name`: Specifies the object name, including all the text that follows the final "`/`".

For example:

```
SELECT genre_id, name, file$name, file$path FROM ext_genre
     WHERE rownum <= 2;

genre_id      name        file$name     file$path
--------      ---------   -----------   ----------------------
1             Action      genre.csv     https://objectstorage.us-
ashburn-1.oraclecloud.com/n/namespace-string/b/moviestream_gold/o/genre
2             Adventure   genre.csv     https://objectstorage.us-
ashburn-1.oraclecloud.com/n/namespace-string/b/moviestream_gold/o/genre
```

See Invisible Columns for more information on invisible columns.

Query External Data with ORC, Parquet, or Avro Source Files

Autonomous Database makes it easy to access ORC, Parquet, or Avro data stored in object store using external tables. ORC, Parquet, and Avro sources have metadata embedded in them and the `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` procedure can utilize this metadata to simplify the creation of external tables.

You don't need to know the structure of the data, `DBMS_CLOUD` can examine the file and convert ORC, Parquet, or Avro contents into the equivalent Oracle columns and data types. You only need to know the location of the data in object store, specify its type, ORC, Parquet, or Avro, and have credentials to access the source file on your object store.

> **Note:**
>
> The steps to use external tables are very similar for ORC, Parquet, and Avro. These steps show working with a Parquet format source file.

The source file in this example, `sales_extended.parquet`, contains Parquet format data. To query this file in Autonomous Database, do the following:

1. Store your object store credentials, to access the object store, using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adb_user@example.com',
    password => 'password' );
END;
/
```

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for creating external tables.

   See CREATE_CREDENTIAL Procedure for information about the `username` and `password` parameters for different object storage services.

2. Create an external table for ORC, Parquet, or Avro on top of your source files using the procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`.

   The procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` supports external files in the supported cloud object storage services, including: Oracle Cloud Infrastructure Object Storage, Azure Blob Storage, Amazon S3, and Amazon S3-Compatible, including: Oracle Cloud Infrastructure Object Storage, Google Cloud Storage, and Wasabi Hot Cloud Storage. The credential is a table level property; therefore, the external files must be on the same object store.

   By default, the columns created in the external table automatically map their data types to Oracle data types for the fields found in the source files and the external table column names match the source field names.

```
BEGIN
   DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
    table_name =>'sales_extended_ext',
    credential_name =>'DEF_CRED_NAME',
    file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/sales_extended.parquet',
    format =>  '{"type":"parquet", "schema": "first"}'
 );
END;
/
```

   The parameters are:

   - `table_name`: is the external table name.
   - `credential_name`: is the name of the credential created in the previous step. The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.

- `file_uri_list`: is a comma delimited list of the source files you want to query.
- `format`: defines the options to describe the format of the source file. For a Parquet file, use the `format` parameter to specify the `type parquet`. For an Avro file use the `format` parameter to specify the `type avro`. For an ORC file use the `format` parameter to specify the `type orc`.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

By default the `format schema` parameter is set and the columns and data types are derived automatically and the fields in the source match the external table columns by name. Source data types are converted to the external table column Oracle data types according to the `DBMS_CLOUD` mapping for ORC, Parquet, or Avro data types. The valid `schema` parameter values are:

- `first`: Analyze the schema of the first ORC, Parquet, or Avro file that `DBMS_CLOUD` finds in the specified `file_uri_list` (`first` is the default value for `schema`).
- `all`: Analyze all the schemas for all the ORC, Parquet, or Avro files found in the `file_uri_list`. Because these are simply files captured in an object store, there is no guarantee that each file's metadata is the same. For example, File1 may contain a field called "address", while File2 may be missing that field. Examining each file to derive the columns is a bit more expensive but may be required if the first file does not contain all the required fields.

> **Note:**
>
> If the `column_list` parameter is specified, then you provide the column names and data types for the external table and the `schema` value, if specified is ignored. Using `column_list` you can limit the columns in the external table. If `column_list` is not specified then the `schema` default value is `first`.

3. You can now run queries on the external table you created in the previous step:

```
DESC sales_extended_ext;
Name           Null? Type
-------------- ----- --------------
PROD_ID              NUMBER(10)
CUST_ID              NUMBER(10)
TIME_ID              VARCHAR2(4000)
CHANNEL_ID           NUMBER(10)
PROMO_ID             NUMBER(10)
QUANTITY_SOLD        NUMBER(10)
AMOUNT_SOLD          NUMBER(10,2)
GENDER               VARCHAR2(4000)
CITY                 VARCHAR2(4000)
STATE_PROVINCE       VARCHAR2(4000)
INCOME_LEVEL         VARCHAR2(4000)
```

```
SELECT prod_id, quantity_sold, gender, city, income_level
     FROM sales_extended_ext
     WHERE ROWNUM < 8;
```

```
     PROD_ID     QUANTITY_SOLD     GENDER    CITY              INCOME_LEVEL
1    13          1                 M         Adelaide          K: 250,000 - 299,999
2    13          1                 M         Dolores           L: 300,000 and above
3    13          1                 M         Cayuga            F: 110,000 - 129,999
4    13          1                 F         Bergen op Zoom    C: 50,000 - 69,999
5    13          1                 F         Neuss             J: 190,000 - 249,999
6    13          1                 F         Darwin            F: 110,000 - 129,999
7    13          1                 M         Sabadell          K:250,000 - 299,999
```

This query shows values for rows in the external table. If you want to query this data frequently, after examining the data you can load it into a table with `DBMS_CLOUD.COPY_DATA`.

See CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files and COPY_DATA Procedure for Avro, ORC, or Parquet Files for more information.

See DBMS_CLOUD URI Formats for information on supported cloud object storage services.

Query External Tables with Partitioning Specified in Source Files

If you want to query multiple data files in the Object Store as a single external table and the files can be represented as multiple logical partitions, it is highly recommended to use an external partitioned table. Using an external partitioned table preserves the logical partitioning of your data files for query access.

Using partitioned external tables has the potential to dramatically improve query performance by only accessing the data required for the query. For example, you may have two years of daily partitions stored in separate objects on Cloud Object Store. When you use partitioned external tables, a query for a single day only needs to access that day's source data. When you use partitioned external tables the database automatically partition prunes, and in this example only needs to scan a very small fraction of the data.

There are two ways to create an external partitioned table with the `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` procedure:

- Using the `file_url_path` value in combination with the `format` parameter: Autonomous Database analyzes Cloud Object Store file path information supplied with this parameter to determine the partition columns and data types (or you can manually specify the partition columns and data types).

  This type of partitioning provides a synchronization routine to handle changes when external partition files are added or removed.

- Using the `partitioning_clause` parameter: Autonomous Database uses the explicit partitioning clause you supply to create an external partitioned table.

  This type of partitioning does not support a synchronization routine.

  See Query External Partitioned Data (with Partitioning Clause) for a description of this type of external table.

- About External Tables with Source File Partitioning
  On Autonomous Database you can create partitioned external tables from Hive style partitioned data or from simple folder partitioned data stored on your Cloud Object Store.
- Query External Partitioned Data with Hive Format Source File Organization
  Use `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` to create an external partitioned table and generate the partitioning information from the Cloud Object Store file path.

- Query External Partitioned Data with Folder Format Source File Organization
  Use `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` to create an external partitioned table and generate the partitioning information from the Cloud Object Store file path.
- Refresh External Partitioned Tables with Updated or Deleted Source Files
  You can use `DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE` to refresh an external partitioned table. Use this procedure when new partitions are added or when partitions are removed from the object store source.

About External Tables with Source File Partitioning

On Autonomous Database you can create partitioned external tables from Hive style partitioned data or from simple folder partitioned data stored on your Cloud Object Store.

Using source file partitioning, instead of supplying a complete partition specification the procedure derives partitioning information from the file path for certain file patterns. For example, consider the following data file specifications:

- Hive style: for example: `sales/country=USA/year=2020/month=01/file1.csv`
- Simple folder partitioning style: for example: `sales/USA/2020/01/file1.parquet`

Using one of these common partitioning formats greatly simplifies both the creation and management of partitioned external tables. In addition, even though partition columns may not appear in the data file, they can still be queried using SQL. Partitioning data also improves query performance by dramatically reducing the amount of data scanned. In this example, when you query 'USA' data, the query can skip scanning the files for other countries.

Hive Format Partitioned Data in Cloud Object Store

Hive offers a standard metadata format for big data processing engines. Partitioned data in Cloud Object Store that is generated in Hive format is represented in a `folder/subfolder` format. For example, on Cloud Object Store a Hive format data file is stored as follows:

```
table/partition1=partition1_value/partition2=partition2_value/data_file.csv
```

Files saved in Hive partitioned format provide partition information in the data file path name. The data file path name includes information about the object contents, including partition column names and partition column values (the data file does not include the partition columns and their associated values).

For example, consider an external partitioned `SALES` table created from Hive format data on Cloud Object Store:

```
.../sales/country=USA/year=2020/month=01/file1.csv
.../sales/country=USA/year=2020/month=01/file2.csv
.../sales/country=USA/year=2020/month=02/file3.csv
.../sales/country=USA/year=2020/month=03/file1.csv
.../sales/country=FRA/year=2020/month=03/file1.csv
```

The Hive format partition information shows the data files in Cloud Object Store are partitioned by `country`, `year`, and `month` and the values for these partition columns are also specified within the Hive format path name for each data file (the path name includes values for the partitioned columns: `country`, `year`, and `month`).

The column names in the path will be used by the API to simplify the table definition.

Simple Folder Format Partitioned Data in Cloud Object Store

Partitioned data in Cloud Object Store that is generated in folder format is represented in a `folder/subfolder` format, similar to Hive format partitioned data, but the information in the path shows the column values and does not include the column names. Also, with folder format partitioned data the partition order specified in the object name is significant, and must match the order in the table columns.

For example, on Cloud Object Store a folder format data file is stored as follows:

```
table/partition1_value/partition2_value/*.parquet
```

The path includes both partition column values, in partition column order, and the data files. Autonomous Database allows you to create an external partitioned table from folder format data and you can perform a query using the specified partitions.

Files saved in folder partitioned format provide the data partition column values in the file name. Unlike Hive, the paths do not include the column name, therefore the column names must be provided. The order of partition columns is important and the order in the file name for column partition names must match the order in the `partition_columns` parameter.

About Querying Partitioned Data in Cloud Object Store

When you query external partitioned data in Hive format, the query engine understands and utilizes the partitioning information from the file path name. For example, consider an external partitioned `SALES` table where the source file, `sales/country=USA/year=2020/month=02/file3.csv` on Object Store includes the following sales data:

```
tents, 291
canoes, 22
backpacks, 378
```

The `country` values in the path name, and the time period values for `month` and `year` are not specified as columns within the data file. The partition column values are specified only in the path name with values shown: USA, 2020, and 02. After you create an external partitioned table with this data file you can use the partition columns and their values when you run a query on the external partitioned table.

For example:

```
SELECT year, month, product, units
     FROM SALES WHERE year='2020' AND month='02' AND country='USA'
```

The benefit of creating an external partitioned table with data generated as Hive format partitioned data is that the query engine is optimized to partition prune the data to select the correct partition and the query only selects data from one partition and only needs to search a single data file. Thus, the query would only require a scan of the `file3.csv` file (`/sales/country=USA/year=2020/month=02/file3.csv`). For large amounts of data, such partition pruning can provide significant performance improvements.

Using standard Oracle Database external tables, the partition column must be available as a column within the data file to use it for queries or partition definitions. Without the special handling that is available with external partitioned tables on Autonomous Database, this would be a problem if you want to use data stored in Hive format on Cloud Object Store, as you would need to regenerate the data files to include the partition as a column in the data file.

About Creating Partitioned External Tables

When you use unstructured data stored in Hive format on Cloud Object Store and you create an external partitioned table, the columns and their types cannot be derived from the source file. Thus, the columns and their data types must be specified with the `column_list` parameter. To create the partitioned external tables, use the procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` to specify the partition columns and their types as follows:

- The root for the file list is specified in the path name with the `file_uri_list` parameter. For example, `http://.../sales/*`
- The column names and data types are specified with the `column_list` parameter.
- The option `partition_columns` in the `format` parameter specifies the partition columns.
- The generated DLL includes the columns specified in the path name.

For this example, when the external table is created the `country`, `year`, and `month` columns are added in the `column_list` parameter. The external table is created with the `country`, `year`, and `month` columns, which are not in the data files, and list partitions are created enabling partition pruning.

When you use structured data, such as Parquet, Avro, or ORC files stored in folder format on Cloud Object Store, the columns and their data types are known. and you do not need to specify the column list as is required with unstructured data. To create the partitioned external tables, use the procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` to specify the partition columns and their types as follows:

- The root for the file list is specified in the path name with the `file_uri_list` parameter. For example, `http://.../sales/*`
- The `column_list` parameter is not required for structured files. If you do not specify the column list, you must define the partition columns and their data types when you create the external partitioned table. Use the option `partition_columns` in the `format` parameter to specify the partition columns and their data types.
- The generated DLL includes the columns specified in the path name.

See Query External Partitioned Data with Hive Format Source File Organization and Query External Partitioned Data with Folder Format Source File Organization for complete examples.

- External Partitioning: CSV Source Files with Hive-style Folders
  Shows how to create external partitioned tables with CSV source files stored on Cloud Object Store in Hive-style folders.
- External Partitioning: CSV Source Files with Simple Folders
  Shows how to create external partitioned tables with CSV source files stored on Cloud Object Store in simple folder format.
- External Partitioning: Parquet Source Files with Hive-style Folders
  Shows how to create external partitioned tables with Parquet source files stored on Cloud Object Store in Hive-style folders.
- External Partitioning: Parquet with Simple Folders
  Shows how to create external partitioned tables with Parquet source files stored on Cloud Object Store in simple folder format.

External Partitioning: CSV Source Files with Hive-style Folders

Shows how to create external partitioned tables with CSV source files stored on Cloud Object Store in Hive-style folders.

Source file list:

```
.../sales/country=USA/year=2020/month=01/file1.csv
.../sales/country=USA/year=2020/month=01/file2.csv
.../sales/country=USA/year=2020/month=02/file3.csv
.../sales/country=USA/year=2020/month=03/file1.csv
.../sales/country=FRA/year=2020/month=03/file1.csv
```

API:

```
DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE (
   table_name        => 'mysales',
   credential_name   => 'mycredential',
   file_uri_list     => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/.../sales/*.csv',
   column_list       => 'product varchar2(100), units number, country
varchar2(100), year number, month varchar2(2)',
   field_list        => 'product, units', --[Because country, year and month
are not in the file, they are not listed in the field list]
   format            => '{"type":"csv","partition_columns":["country",
"year", "month"]}');
```

> **Note:**
>
> The `partition_columns` in the `format` parameter must match the column names found in the path (for example, the `country` column matches "`country=…`")

External Partitioning: CSV Source Files with Simple Folders

Shows how to create external partitioned tables with CSV source files stored on Cloud Object Store in simple folder format.

Source file list:

```
.../sales/USA/2020/01/file1.csv
.../sales/USA/2020/01/file2.csv
.../sales/USA/2020/02/file3.csv
.../sales/USA/2020/03/file1.csv
.../sales/FRA/2020/03/file1.csv
```

API:

```
DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
   table_name        => 'mysales',
   credential_name   => 'mycredential',
   file_uri_list     =>  'https://objectstorage.us-
phoenix-1.oraclecloud.com/.../sales/*.csv',
```

```
   column_list       => 'product varchar2(100), units number, country
varchar2(100), year number, month varchar2(2)',
   field_list        => 'product, units', --[Because country, year and month
are not in the file, they are not listed in the field list]
   format            => '{"type":"csv","partition_columns":["country",
"year", "month"]}');
```

> **Note:**
>
> The API call is the same as in the previous example, but the order of the `partition_columns` in the `format` parameter is significant because the column name is not in the file path.

External Partitioning: Parquet Source Files with Hive-style Folders

Shows how to create external partitioned tables with Parquet source files stored on Cloud Object Store in Hive-style folders.

Source file list:

```
.../sales/USA/2020/01/file1.parquet
.../sales/USA/2020/01/file2.parquet
.../sales/USA/2020/02/file3.parquet
.../sales/USA/2020/03/file1.parquet
.../sales/FRA/2020/03/file1.parquet
```

API:

```
DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE (
   table_name            => 'mysales',
     credential_name     => 'mycredential',
     file_uri_list       => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/.../sales/*.parquet',
     format              =>
         json_object(    'type' value 'parquet',
               'schema' value 'first',
               'partition_columns' value
                  json_array(
                      json_object('name' value 'country', 'type' value
'varchar2(100)'),
                      json_object('name' value 'year', 'type' value 'number'),
                      json_object('name' value 'month', 'type' value
'varchar2(2)')
            )
        )

);
```

> **Note:**
>
> The `column_list` parameter is not specified. As shown, for each partition column specify both the name and data type in the `format` parameter `partition_columns`.

External Partitioning: Parquet with Simple Folders

Shows how to create external partitioned tables with Parquet source files stored on Cloud Object Store in simple folder format.

Source file list:

```
.../sales/USA/2020/01/file1.parquet
.../sales/USA/2020/01/file2.parquet
.../sales/USA/2020/02/file3.parquet
.../sales/USA/2020/03/file1.parquet
.../sales/FRA/2020/03/file1.parquet
```

API:

```
DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE (
   table_name        => 'mysales',
   credential_name   => 'mycredential',
   file_uri_list     => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/.../sales/*.parquet',
   format            =>
     json_object(    'type' value 'parquet',
                 'schema' value 'first',
          'partition_columns' value
            json_array(
               json_object('name' value 'country', 'type' value
'varchar2(100)'),
               json_object('name' value 'year', 'type' value 'number'),
               json_object('name' value 'month', 'type' value 'varchar2(2)')
             )
        )

);
```

> **Note:**
>
> The `column_list` parameter is not specified. You must include both the name and data type for the partition columns. In addition, the order of the `partition_columns` in the format clause matters because the column name is not in the file path.

Query External Partitioned Data with Hive Format Source File Organization

Use `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` to create an external partitioned table and generate the partitioning information from the Cloud Object Store file path.

Consider the following sample source files in Object Store:

```
custsales/month=2019-01/custsales-2019-01.csv

custsales/month=2019-02/custsales-2019-02.csv

custsales/month=2019-03/custsales-2019-03.csv
```

With this naming, the values for `month` are captured within the object name.

To create a partitioned external table with data stored in this sample Hive format, do the following:

1. Store Object Store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL (
       credential_name => 'DEF_CRED_NAME',
       username => 'adb_user@example.com',
       password => 'password' );
   END;
   /
   ```

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for creating external tables.

   See CREATE_CREDENTIAL Procedure for information about the `username` and `password` parameters for different object storage services.

2. Create an external partitioned table on top of your source files using the procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`.

   The procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` supports external partitioned files in the supported cloud object storage services. The credential is a table level property; therefore, the external files must all be on the same cloud object store.

   For example:

   ```
   BEGIN
       DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
           TABLE_NAME => 'sales_sample',
           CREDENTIAL_NAME => 'DEF_CRED_NAME',
           FILE_URI_LIST => 'https://objectstorage.us-
   ashburn-1.oraclecloud.com/n/namespace-string/b/moviestream_landing/o/
   sales_sample/*.parquet',
           FORMAT => '{"type":"parquet", "schema":
   "first","partition_columns":[{"name":"month","type":"varchar2(100)"}]}');
   END;
   /
   ```

   The parameters are:

   - `table_name`: is the external table name.

- `credential_name`: is the name of the credential created in the previous step.
- `file_uri_list`: is a comma-delimited list of source file URIs. There are two options for this list:
  - Specify a comma-delimited list of individual file URIs without wildcarding.
  - Specify a single file URI with wildcards, where the wildcards can only be after the last slash "/". The character "*" can be used as the wildcard for multiple characters, the character "?" can be used as the wildcard for a single character.
- `column_list`: is a comma delimited list of column names and data types for the external table. The list includes the columns inside the data file and those derived from the object name (from names in the file path).

  The `column_list` is not required when the data files are structured files (Parquet, Avro, or ORC).
- `format`: defines the options you can specify to describe the format of the source file. The `partition_columns format` parameter specifies the names of the partition columns.

  If the data in your source file is encrypted, decrypt the data by specifying the `encryption` format option. See Decrypt Data While Importing from Object Storage for more information on decrypting data.

  See DBMS_CLOUD Package Format Options for more information.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

The `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` call would result in the following table definition:

```
CREATE TABLE "ADMIN"."SALES_SAMPLE"
   ( "DAY_ID" TIMESTAMP (6),
      "GENRE_ID" NUMBER(19,0),
      "MOVIE_ID" NUMBER(19,0),
      "CUST_ID" NUMBER(19,0),
      "APP" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
      "DEVICE" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
      "OS" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
      "PAYMENT_METHOD" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
      "LIST_PRICE" BINARY_DOUBLE,
      "DISCOUNT_TYPE" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
      "DISCOUNT_PERCENT" BINARY_DOUBLE,
      "ACTUAL_PRICE" BINARY_DOUBLE,
      "MONTH" VARCHAR2(100 BYTE) COLLATE "USING_NLS_COMP"
   ) DEFAULT COLLATION "USING_NLS_COMP"
   ORGANIZATION EXTERNAL
    ( TYPE ORACLE_BIGDATA
      DEFAULT DIRECTORY "DATA_PUMP_DIR"
      ACCESS PARAMETERS
      ( com.oracle.bigdata.fileformat=parquet
com.oracle.bigdata.filename.columns=["month"]
com.oracle.bigdata.file_uri_list="https://objectstorage.us-
ashburn-1.oraclecloud.com/n/namespace-string/b/moviestream_landing/o/
sales_sample/*.parquet"
com.oracle.bigdata.credential.schema="ADMIN"
```

```
com.oracle.bigdata.credential.name=CRED_OCI
com.oracle.bigdata.trimspaces=notrim
  )
    )
   REJECT LIMIT 0
  PARTITION BY LIST ("MONTH")
 (PARTITION "P1" VALUES (('2019-01'))
      LOCATION
       ( 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-
string/b/moviestream_landing/o/sales_sample/month=2019-01/*.parquet'
       ),
 PARTITION "P2" VALUES (('2019-02'))
      LOCATION
       ( 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-
string/b/moviestream_landing/o/sales_sample/month=2019-02/*.parquet'
       ))
  PARALLEL ;
```

See CREATE_EXTERNAL_PART_TABLE Procedure for detailed information about the parameters.

See DBMS_CLOUD URI Formats for more information on the supported cloud object storage services.

3. You can now run queries on the external partitioned table you created in the previous step.

   Your Autonomous Database takes advantage of the partitioning information of your external partitioned table, ensuring that the query only accesses the relevant data files in the Object Store.

   For example:

   ```
   SELECT movie_id, month FROM sales WHERE month='2019-02'
   ```

   The external partitioned tables you create with `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from. See External Table Metadata Columns for more information.

If there are any rows in the source files that do not match the format options you specified, the query reports an error. You can use `DBMS_CLOUD` parameters, like *rejectlimit,* to suppress these errors. As an alternative, you can also validate the external partitioned table you created to see the error messages and the rejected rows so that you can change your format options accordingly. See Validate External Data and Validate External Partitioned Data for more information.

Query External Partitioned Data with Folder Format Source File Organization

Use `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` to create an external partitioned table and generate the partitioning information from the Cloud Object Store file path.

When you create an external table with folder format data files, you have two options for specifying the types of the partition columns:

- You can manually specify the columns and their data types with `column_list` parameter. See Query External Partitioned Data with Hive Format Source File Organization for an example using the `column_list` parameter.

- You can let `DBMS_CLOUD` derive the data file columns and their types from information in structured data files such as Avro, ORC, and Parquet data files. In this case, you use the `partition_columns` option with the `format` parameter to supply the column names and their data types for the partition columns and you do not need to supply the `column_list` or the `field_list` parameters.

Consider the following sample source files in Object Store:

```
.../sales/USA/2020/01/sales1.parquet

.../sales/USA/2020/02/sales2.parquet
```

To create a partitioned external table with the Cloud Object Store file path defining the partitions from files with this sample folder format, do the following:

1. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL (
       credential_name => 'DEF_CRED_NAME',
       username => 'adb_user@example.com',
       password => 'password' );
   END;
   /
   ```

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for creating external tables.

   See CREATE_CREDENTIAL Procedure for information about the `username` and `password` parameters for different object storage services.

2. Create an external partitioned table on top of your source files using the procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`.

   The procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` supports external partitioned files in the supported cloud object storage services. The credential is a table level property; therefore, the external files must all be on the same cloud object store.

   For example:

```
BEGIN DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
    table_name => 'MYSALES',
    credential_name => 'DEF_CRED_NAME',
    file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/sales/*.parquet',
    format =>
        json_object('type' value 'parquet', 'schema' value 'first',
                    'partition_columns' value
                          json_array(
                                json_object('name' value 'country', 'type' value
'varchar2(100)'),
                                json_object('name' value 'year', 'type' value 'number'),
```

```
                                    json_object('name' value 'month', 'type' value
'varchar2(2)')
                                )
          )
    );
END;
/
```

The `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` parameters for structured data files, such as for a Parquet data file does not require the `column_list` or the `field_list` parameters. The column names and data types are derived for the columns from the first parquet file that the procedure scans (and therefore all files must have the same shape). The generated column list includes the columns derived from the object name and these column have the data types specified with the `partition_columns format` parameter.

The parameters are:

- `table_name`: is the external table name.
- `credential_name`: is the name of the credential created in the previous step.
- `file_uri_list`: is a comma-delimited list of source file URIs. There are two options for this list:
  - Specify a comma-delimited list of individual file URIs without wildcarding.
  - Specify a single file URI with wildcards, where the wildcards can only be after the last slash "/". The character "*" can be used as the wildcard for multiple characters, the character "?" can be used as the wildcard for a single character.
- `column_list`: is a comma delimited list of column names and data types for the external table. The list includes the columns that are inside the file as well as those derived from the object name.

  The `column_list` is not required when the data files are structured files (Parquet, Avro, or ORC).
- `field_list`: Identifies the fields in the source files and their data types. The default value is `NULL` meaning the fields and their data types are determined by the `column_list` parameter.

  The `field_list` is not required when the data files are structured files (Parquet, Avro, or ORC).
- `format`: defines the options you can specify to describe the format of the source file. The `partition_columns format` parameter specifies the names of the partition columns. See DBMS_CLOUD Package Format Options for more information.

  If the data in your source file is encrypted, decrypt the data by specifying the `encryption` format option. See Decrypt Data While Importing from Object Storage for more information on decrypting data.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

See CREATE_EXTERNAL_PART_TABLE Procedure for detailed information about the parameters.

See DBMS_CLOUD URI Formats for more information on the supported cloud object storage services.

If there are any rows in the source files that do not match the format options you specified, the query reports an error. You can use `DBMS_CLOUD` parameters, like `rejectlimit` to suppress these errors. As an alternative, you can also validate the external partitioned table you created to see the error messages and the rejected rows so that you can change your format options accordingly. See Validate External Data and Validate External Partitioned Data for more information.

3. You can now run queries on the external partitioned table you created in the previous step.

   Your Autonomous Database takes advantage of the partitioning information of your external partitioned table, ensuring that the query only accesses the relevant data files in the Object Store. For example, the following query only reads data files from one partition.

   For example:

   ```
   SELECT year, month, product, units
   FROM SALES WHERE year='2020' AND month='02' AND country='USA'
   ```

   The external partitioned tables you create with `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from. See External Table Metadata Columns for more information.

Refresh External Partitioned Tables with Updated or Deleted Source Files

You can use `DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE` to refresh an external partitioned table. Use this procedure when new partitions are added or when partitions are removed from the object store source.

> **Note:**
>
> Only use `DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE` when the partitioned external table is created with `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` and the `file_url_path` parameter.

To refresh a partitioned external table:

1. Refresh an external partitioned table on top of your Cloud Object Store source files using the procedure `DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE`.

   For example:

```
BEGIN
   DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE(table_name => 'MYSALES');
END;
/
```

   `DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE` uses the credential information from the corresponding `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` with the specified `table_name` to access Cloud Object Store.

   See SYNC_EXTERNAL_PART_TABLE Procedure for detailed information about the parameter.

2. The partition information is now refreshed and you can run queries on the updated external partitioned table. with new partitions available or with partitions that were removed no longer available in the external partitioned table.

Query External Partitioned Data (with Partitioning Clause)

If you want to query multiple data files in the Object Store as a single external table and the files can be represented as multiple logical partitions, then it is highly recommended to use an external partitioned table. Using an external partitioned table preserves the logical partitioning of your data files for query access. Use the procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` to create an external partitioned table.

There are two ways to create an external partitioned table on Autonomous Database:

- The first version is for. See Query External Partitioned Data with Hive Format Source File Organization for information on this type of external partitioned table usage.
- The second version is for. When you a create a partitioned external table in this way you include a partitioning clause in the `partitioning_clause` parameter. The partitioning clause that you include depends upon the data files in the Cloud and the type of partitioning you use. This section describes this type of external partitioned table usage.

1. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL (
       credential_name => 'DEF_CRED_NAME',
       username => 'adb_user@example.com',
       password => 'password' );
   END;
   /
   ```

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for creating external tables.

   See CREATE_CREDENTIAL Procedure for information about the `username` and `password` parameters for different object storage services.

2. Create an external partitioned table on top of your source files using the procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`.

   The procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` supports external partitioned files in the supported cloud object storage services. The credential is a table level property; therefore, the external files must be on the same object store.

   For example:

```
BEGIN
  DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
      table_name =>'PET1',
      credential_name =>'DEF_CRED_NAME',
```

```
      format => json_object('delimiter' value ',', 'recorddelimiter' value 'newline',
'characterset' value 'us7ascii'),
      column_list => 'col1 number, col2 number, col3 number',
      partitioning_clause => 'partition by range (col1)
         (partition p1 values less than (1000) location
             ( ''https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o/file_11.txt''),
          partition p2 values less than (2000) location
             ( ''https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o/file_21.txt''),
          partition p3 values less than (3000) location
             ( ''https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o/file_31.txt'') )'
     );
END;
/
```

The parameters are:

- `table_name`: is the external table name.
- `credential_name`: is the name of the credential created in the previous step.
- `partitioning_clause`: is the complete partitioning clause, including the location information for individual partitions.
- `format`: defines the options you can specify to describe the format of the source file. format: defines the options you can specify to describe the format of the source file.

  If the data in your source files is encrypted, decrypt the data by specifying the `format` parameter with the `encryption` option. See Decrypt Data While Importing from Object Storage for more information on decrypting data.
- `column_list`: is a comma delimited list of the column definitions in the source files.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

You can now run queries on the external partitioned table you created in the previous step. Your Autonomous Database takes advantage of the partitioning information of your external partitioned table, ensuring that the query only accesses the relevant data files in the Object Store. For example, the following query only reads data files from partition P1:

```
SELECT * FROM pet1 WHERE col1 < 750;
```

The external partitioned tables you create with `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from. See External Table Metadata Columns for more information.

If there are any rows in the source files that do not match the format options you specified, the query reports an error. You can use `DBMS_CLOUD` parameters, like *rejectlimit,* to suppress these errors. As an alternative, you can also validate the external partitioned table you created to see the error messages and the rejected rows so that you can change your format options accordingly. See Validate External Data and Validate External Partitioned Data for more information.

See CREATE_EXTERNAL_PART_TABLE Procedure for detailed information about the parameters.

See DBMS_CLOUD URI Formats for more information on the supported cloud object storage services.

Query Hybrid Partitioned Data

If you want to query internal data and multiple data files in the Object Store as single logical table you can use a hybrid partitioned table to represent the data as single object. Use the procedure `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` to create a hybrid partitioned table.

If your data, internal or external, can be represented in finer granularity as multiple logical partitions then it is highly recommended to create a hybrid partitioned table with multiple internal and external partitions, preserving the logical partitioning of your data for query access.

When you a create a hybrid partitioned table, you include a partitioning clause in the `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` statement. The partitioning clause that you include depends upon your data files and the type of partitioning you use. See Creating Hybrid Partitioned Tables for more information.

1. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL (
       credential_name => 'DEF_CRED_NAME',
       username => 'adb_user@example.com',
       password => 'password' );
   END;
   /
   ```

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for creating hybrid partitioned tables.

   See CREATE_CREDENTIAL Procedure for information about the `username` and `password` parameters for different object storage services.

2. Create a hybrid partitioned table on top of your source files using the procedure `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE`.

   The procedure `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` supports external partitioned files in the supported cloud object storage services. The credential is a table level property; therefore, the external files must be on the same object store.

   For example:

```
BEGIN
  DBMS_CLOUD.CREATE_HYBRID_PART_TABLE(
      table_name =>'HPT1',
      credential_name =>'DEF_CRED_NAME',
      format => json_object('delimiter' value ',', 'recorddelimiter' value 'newline',
```

```
'characterset' value 'us7ascii'),
      column_list => 'col1 number, col2 number, col3 number',
      partitioning_clause => 'partition by range (col1)
           (partition p1 values less than (1000) external location
              ( ''https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o/file_11.txt'') ,
            partition p2 values less than (2000) external location
              ( ''https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o/file_21.txt'') ,
            partition p3 values less than (3000) )'
     );
END;
/
```

The parameters are:

- `table_name`: is the hybrid partitioned table name.
- `credential_name`: is the name of the credential created in the previous step.
- `partitioning_clause`: is the complete partitioning clause, including the location information for individual partitions.
- `format`: defines the options you can specify to describe the format of the source file.

  If the data in your source files is encrypted, decrypt the data by specifying the `format` parameter with the `encryption` option. See Decrypt Data While Importing from Object Storage for more information on decrypting data.
- `column_list`: is a comma delimited list of the column definitions in the source files.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

You can now run queries on the hybrid partitioned table you created in the previous step. Your Autonomous Database takes advantage of the partitioning information of your hybrid partitioned table, ensuring that the query only accesses relevant data files in the Object Store. For example, the following query only reads data files from partition P1:

```
SELECT * FROM hpt1 WHERE col1 < 750;
```

The hybrid partitioned tables you create with `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from. See External Table Metadata Columns for more information.

If there are any rows in the source files that do not match the format options you specified, the query reports an error. You can use `DBMS_CLOUD` parameters, like *rejectlimit,* to suppress these errors. As an alternative, you can also validate the hybrid partitioned table you created to see the error messages and the rejected rows so that you can change your format options accordingly. See Validate External Data and Validate Hybrid Partitioned Data for more information.

See CREATE_HYBRID_PART_TABLE Procedure detailed information about the parameters.

See DBMS_CLOUD URI Formats for more information on the supported cloud object storage services.

Query External Data Pump Dump Files

You can also query Oracle Data Pump dump files in the Cloud by creating an external table using `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`.

The source files to create this type of external table must be exported from the source system using the `ORACLE_DATAPUMP` access driver in External Tables. See Unloading and Loading Data with the ORACLE_DATAPUMP Access Driver for details on exporting using the `ORACLE_DATAPUMP` access driver.

To create an external table you first move the Oracle Data Pump dump files that were exported using the `ORACLE_DATAPUMP` access driver to your Object Store and then use `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` to create the external table.

The source files in this example are the Oracle Data Pump dump files, `exp01.dmp` and `exp02.dmp`.

1. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DEF_CRED_NAME',
       username => 'adb_user@example.com',
       password => 'password' );
   END;
   /
   ```

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for creating external tables.

   See CREATE_CREDENTIAL Procedure for information about the `username` and `password` parameters for different object storage services.

2. Create an external table on top of your source files using the procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`.

   For example:

   ```
   BEGIN
      DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
       table_name =>'CHANNELS_EXT',
       credential_name =>'DEF_CRED_NAME',
       file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
   namespace-string/b/bucketname/o/exp01.dmp,
                        https://objectstorage.us-phoenix-1.oraclecloud.com/n/
   namespace-string/b/bucketname/o/exp02.dmp'
       format => json_object('type' value 'datapump', 'rejectlimit' value
   '1'),
       column_list => 'CHANNEL_ID NUMBER, CHANNEL_DESC VARCHAR2(20),
   CHANNEL_CLASS VARCHAR2(20)' );
   END;
   /
   ```

The parameters are:

- `table_name`: is the external table name.
- `credential_name`: is the name of the credential created in the previous step.
- `file_uri_list`: is a comma delimited list of the Data Pump dump files you want to query.
- `format`: defines the options you can specify to describe the format of the source file defines the options you can specify to describe the format of the source file. When you specify the type `'datapump'`, the only other valid format parameter is `'rejectlimit'`.
- `column_list`: is a comma delimited list of the column definitions in the source files.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

You can now run queries on the external table you created in the previous step. For example:

```
SELECT count(*) FROM channels_ext;
```

By default the database expects all rows in the external data file to be valid and match both the target data type definitions as well as the format definition of the files. As part of validation, `DBMS_CLOUD` makes sure all the necessary dump file parts are there and also checks that the dump files are valid and not corrupt (for example `exp01.dmp`, `exp02.dmp`, and so on). You can use the `DBMS_CLOUD` format option `rejectlimit` to suppress these errors. As an alternative, you can also validate the external table you created to see the error messages and the rejected rows. See Validate External Data for more information.

For detailed information about the parameters, see CREATE_EXTERNAL_TABLE Procedure.

See DBMS_CLOUD URI Formats for more information on the supported cloud object storage services.

Query Big Data Service Hadoop (HDFS) Data from Autonomous Database

You can create database links to Oracle Big Data Service from Autonomous Database.

Big Data Service provides enterprise-grade Hadoop as a service, with end-to-end security, high performance, and ease of management and upgradeability. After deploying the Oracle Cloud SQL Query Server to Big Data Service, you can easily query data available on Hadoop clusters at scale from Autonomous Database using SQL. This allows you to blend data coming from your data lake with data in your Autonomous Database.

Oracle Cloud SQL Query Server is an Oracle SQL engine that can be accessed using database links in Autonomous Database. You create links to Big Data Service using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. See Integrating Cloud SQL with Autonomous Database for more information.

Query External Data with Data Catalog

Oracle Cloud Infrastructure Data Catalog is the metadata management service for Oracle Cloud that helps you discover data and support data governance. It provides an inventory of assets, a business glossary, and a common metastore for data lakes.

Autonomous Database can leverage this metadata to dramatically simplify management for access to your data lake's object store. Instead of manually defining external tables to access your data lake, use the external tables that are defined and managed automatically. These tables will be found in Autonomous Database protected schemas that are kept up to date with changes in Data Catalog.

For more information about Data Catalog, please refer to Data Catalog documentation.

- About Querying with Data Catalog
  By synchronizing with Data Catalog metadata, Autonomous Database automatically creates external tables for each logical entity harvested by Data Catalog. These external tables are defined in database schemas that are fully managed by the metadata synchronization process. Users can immediately query data without having to manually derive the schema (columns and data types) for external data sources and manually create external tables.
- Concepts Related to Querying with Data Catalog
  An understanding of the following concepts is necessary for querying with Data Catalog.
- Synchronization Mapping
  The synchronization process creates and updates Autonomous Database schemas and external tables based on Data Catalog data assets, folders, logical entities, attributes and relevant custom overrides.
- Typical Workflow with Data Catalog
  There is a typical workflow of actions performed by users who want to query with Data Catalog.
- Example: MovieStream Scenario
  In this scenario, Moviestream is capturing data in a landing zone on object storage. Much of this data, but not necessarily all, is then used to feed an Autonomous Database. Prior to feeding Autonomous Database, the data is transformed, cleansed and subsequently stored in the "gold" area.
- Example: Partitioned Data Scenario
  This scenario illustrates how to create external tables in Autonomous Database that are based on Data Catalog logical entities harvested from partitioned data in Object Store.

About Querying with Data Catalog

By synchronizing with Data Catalog metadata, Autonomous Database automatically creates external tables for each logical entity harvested by Data Catalog. These external tables are defined in database schemas that are fully managed by the metadata synchronization process. Users can immediately query data without having to manually derive the schema (columns and data types) for external data sources and manually create external tables.

Synchronization is dynamic, keeping the Autonomous Database up-do-date with respect to changes to the underlying data, reducing administration cost as it automatically maintains hundreds to thousands of tables. It also allows multiple Autonomous Database instances to share the same Data Catalog, further reducing management costs and providing a common set of business definitions.

The Data Catalog folders/buckets are containers that sync with Autonomous Database schemas. Logical entities within those folders/buckets map to Autonomous Database external tables. These schemas and external tables are automatically generated and maintained through the sync process:

- Folders/Buckets map to database schemas that are for organizational purposes only.
- The organization is meant to be consistent with the data lake and minimize confusion when accessing data thru different paths.

- Data Catalog is the source of truth for the tables contained within schemas. Changes made in the Data Catalog update the schema's tables during a subsequent sync.

To use this capability, a Database Data Catalog Administrator initiates a connection to a Data Catalog instance, selects which data assets and logical entities to synchronize, and runs the sync. The sync process creates schemas and external tables based on the selected Data Catalog harvested data assets and logical entities. As soon as the external tables are created, Data Analysts can start querying their data without having to manually derive the schema for external data sources and create external tables.

> **Note:**
>
> The `DBMS_DCAT` Package is available for performing the tasks required to query Data Catalog object store data assets. See DBMS_DCAT Package.

Concepts Related to Querying with Data Catalog

An understanding of the following concepts is necessary for querying with Data Catalog.

Data Catalog
Data Catalog harvests data assets that point to the object store data sources you want to query with Autonomous Database. From Data Catalog you can specify how the data is organized during harvesting, supporting different file organization patterns. As part of the Data Catalog harvesting process, you can select the buckets and files you want to manage within the asset. For further information, see Data Catalog Overview.

Object Stores
Object Stores have buckets containing a variety of objects. Some common types of objects found in these buckets include: CSV, parquet, avro, json, and ORC files. Buckets generally have a structure or a design pattern to the objects they contain. There are many different ways to structure data and many different ways of interpreting these patterns.
For example, a typical design pattern uses top-level folders that represent tables. Files within a given folder share the same schema and contain data for that table. Subfolders are often used to represent table partitions (for example, a subfolder for each day). Data Catalog refers to each top-level folder as a logical entity, and this logical entity maps to an Autonomous Database external table.

Connection
A connection is an Autonomous Database connection to a Data Catalog instance. For each Autonomous Database instance there can be connections to multiple Data Catalog instances. The Autonomous Database credential must have rights to access Data Catalog assets that have been harvested from object storage.

Harvest
A Data Catalog process that scans object storage and generates the logical entities from your data sets.

Data Asset
A data asset in Data Catalog represents a data source, which includes databases, Oracle Object Storage, Kafka, and more. Autonomous Database leverages Oracle Object Storage assets for metadata synchronization.

Data Entity
A data entity in Data Catalog is a collection of data such as a database table or view, or a single file and normally has many attributes that describe its data.

Logical Entity
In Data Lakes, numerous files typically comprise a single logical entity. For example, you may have daily clickstream files, and these files share the same schema and file type.
A Data Catalog logical entity is a group of Object Storage files that are derived during harvesting by applying filename patterns that have been created and assigned to a data asset.

Data Object
A data object in Data Catalog refers to data assets and data entities.

Filename Pattern
In a data lake, data may be organized in different ways. Typically, folders capture files of the same schema and type. You must register to Data Catalog how your data is organized. Filename patterns are used to identify how your data is organized. In Data Catalog, you can define filename patterns using regular expressions. When Data Catalog harvests a data asset with an assigned filename pattern, logical entities are created based on the filename pattern. By defining and assigning these patterns to data assets, multiple files can be grouped as logical entities based on the filename pattern.

Synchronize (Sync)
Autonomous Database performs synchronizations with Data Catalog to automatically keep its database up-to-date with respect to changes to the underlying data. Synchronization can be performed manually, or on a schedule.
The sync process creates schemas and external tables based on the Data Catalog data assets and logical entities. These schemas are protected, which means their metadata is managed by Data Catalog. If you want to alter the metadata, you must make the changes in Data Catalog. The Autonomous Database schemas will reflect any changes after the next sync is run. For further details, see Synchronization Mapping.

Synchronization Mapping

The synchronization process creates and updates Autonomous Database schemas and external tables based on Data Catalog data assets, folders, logical entities, attributes and relevant custom overrides.

Data Catalog	Autonomous Database	Mapping Description
Data asset and folder (object storage bucket)	Schema name	**Default values**: By default, the generated schema name in Autonomous Database has the following format: `DCAT$<dcat-con-id>_<data-asset-name>_<folder-name>` • `dcat-con-id` is the unique Data Catalog connection identifier. For details on specifying this identifier, see the `dcat_con_id` parameter in the DBMS_DCAT RUN_SYNC Procedure. • `data-asset-name` is the name of the Data Catalog data asset's name. • `folder-name` is the Data Catalog folder name. This folder maps to an object storage bucket. **Customizations**: The default `data-asset-name` and `folder-name` can be customized by defining custom properties, business names and display names to override these default names. • `data-asset-name` can be overidden by defining the `oracle-db-schema-prefix` custom property for the data asset in Data Catalog. • `folder-name` can be overidden by defining the `oracle-db-schema` custom property for the folder in Data Catalog, a business name or display name. The following attributes are used in order of precedence for generating the `folder-name`: 1. `oracle-db-schema` custom property 2. Business Name 3. Display Name **Examples**: • If the connection id is `DataModels`, the asset name is `ObjectStorage`, the folder name is `HR` and there are no custom property overrides, the derived schema name is: `DCAT$DATAMODELS_OBJECTSTORAGE_HR` • If the data asset name is MYASSET, the folder name is MYFOLDER, and there are no custom property overrides, the schema name is: `DCAT$MYASSET_MYFOLDER` • If the data asset has `oracle-db-prefix = FIRSTASSET`, and the folder has `oracle-db-schema = FIRSTFOLDER`, then the schema name is: `DCAT$FIRSTASSET_FIRSTFOLDER`

Data Catalog	Autonomous Database	Mapping Description
Logical entity	External table	Logical entities are mapped to external tables. If the logical entity has a partitioned attribute, it is mapped to a partitioned external table. The external table name is derived from the corresponding logical entity's Display Name, or Business Name. If `oracle-db-schema` is set, then its value overrides all the names and custom properties of the corresponding folders and data assets. For example, if `oracle-db-schema` for an entity is set to `EntitySchema`, then the table is created in schema `DCAT$ENTITYSCHEMA`.
Logical entity's attributes	External table columns	**Column names**: The external table column names are derived from the corresponding logical entity's attribute display names, or business names. For logical entities derived from Parquet, Avro, and ORC files, the column name is always the display name of the attribute as it represents the field name derived from the source files. For attributes corresponding to a logical entity derived from CSV files, the following attribute fields are used in order of precedence for generating the column name: 1. `oracle-db-column-name` 2. Business Name 3. Display Name **Column type**: The `oracle-db-column-type` custom property overrides the default column type that was derived by Data Catalog. For attributes corresponding to a logical entity derived from Avro files with `TIME_MICROS`, `TIME_MILLIS`, `TIMESTAMP_MICROS` or `TIMESTAMP_MILLIS` data types, you must set the `oracle-db-column-type` of the corresponding attribute in Data Catalog. **Column length**: The `oracle-db-column-length` custom property overrides the default column length for a string field that was derived by Data Catalog. **Column precision**: The `oracle-db-column-precision` custom property overrides the default precision for a number that was derived by Data Catalog. For attributes corresponding to a logical entity derived from Avro files with `TIME_MICROS`, `TIME_MILLIS`, `TIMESTAMP_MICROS` or `TIMESTAMP_MILLIS` data types, you must set the `oracle-db-column-precision` of the corresponding attribute in Data Catalog. **Column scale**: The `oracle-db-column-scale` custom property overrides the default scale for a number that was derived by Data Catalog.

Typical Workflow with Data Catalog

There is a typical workflow of actions performed by users who want to query with Data Catalog.

The Database Data Catalog Admin creates a connection between the Autonomous Database instance and a Data Catalog instance, then configures and runs a synchronization (sync) between the Data Catalog and Autonomous Database. The sync creates external tables and schemas in the Autonomous Database instance based on the synced Data Catalog contents. The Database Data Catalog Query Admin or Database Admin grants READ access to the generated external tables so that Data Analysts and other database users can browse and query the external tables.

The table below describes each action in detail. For a description of the different user types included in this table, see Data Catalog Users and Roles.

> **Note:**
>
> The `DBMS_DCAT` Package is available for performing the tasks required to query Data Catalog object store data assets. See DBMS_DCAT Package.

Action	Who is the user	Description
Create policies	Database Data Catalog Administrator	The Autonomous Database resource principal or Autonomous Database user credential must have the appropriate permissions to manage Data Catalog and to read from object storage. **More information**: Required Credentials and IAM Policies.
Create credentials	Database Data Catalog Administrator	Ensure database credentials are in place to access Data Catalog and to query object store. The user calls `DBMS_CLOUD.CREATE_CREDENTIAL` to create user credentials and/or `DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL` to enable resource principals. **More information**: DBMS_CLOUD CREATE_CREDENTIAL Procedure, Use Resource Principal with DBMS_CLOUD.

Action	Who is the user	Description
Create connections to Data Catalog	Database Data Catalog Administrator	To initiate a connection between an Autonomous Database instance and a Data Catalog instance the user calls `DBMS_DCAT.SET_DATA_CATALOG_CONN` to specify the target Data Catalog instance. Connections from an Autonomous Database instance to multiple Data Catalog instances is supported. The connection to the Data Catalog instance must use a database credential object with sufficient Oracle Cloud Infrastructure (OCI) privileges. For example, the Resource Principal Service Token for the Autonomous Database instance or an OCI user with sufficient privileges can be used. Once the connection has been made, the Data Catalog instance is updated with the `DBMS_DCAT` namespace and custom properties (if they do not already exist). The user can run a query to see the new connection including all current connections: `select * from all_dcat_connections;` **More information**: SET_DATA_CATALOG_CONN Procedure, UNSET_DATA_CATALOG_CONN Procedure.
Create a selective sync	Database Data Catalog Administrator	Create a sync job by selecting the Data Catalog objects to sync. The user can: • Select data assets/folders to sync. • Select individual logical entities to sync. • Preview the resulting external tables before syncing. • Change external tables (for example, the name) by modifying custom properties on Data Catalog. **More information**: See CREATE_SYNC_JOB Procedure, DROP_SYNC_JOB Procedure, Synchronization Mapping
Sync with Data Catalog	Database Data Catalog Administrator	The user initiates a sync operation. The sync is initiated manually through the `DBMS_DCAT.RUN_SYNC` procedure call, or automatically as part of a scheduled sync job. The sync operation creates, modifies and drops external tables and schemas according to the Data Catalog contents and sync selections. Manual configuration is applied using Data Catalog Custom Properties. **More information**: See RUN_SYNC Procedure, CREATE_SYNC_JOB Procedure, Synchronization Mapping
Monitor sync and view logs	Database Data Catalog Administrator	The user can view the sync status by querying the `USER_LOAD_OPERATIONS` view. After the sync process has completed, the user can view a log of the sync results, including details about the mappings of logical entities to external tables. **More information**: Monitoring and Troubleshooting Loads
Grant privileges	Database Data Catalog Query Administrator, Database Administrator	The database Data Catalog Query Administrator or database Administrator must grant READ on generated external tables to data analyst users. This allows the data analysts to query the generated external tables.

Action	Who is the user	Description
Browse and query external tables	Data Analyst	Data analysts are able to query the external tables through any tool or application that supports Oracle SQL. Data Analysts can review the synced schemas and tables in the DCAT$* schemas, and query the tables using Oracle SQL. **More information**: Synchronization Mapping
Terminate connections to Data Catalog	Database Data Catalog Administrator	To remove an existing Data Catalog association, the user calls the UNSET_DATA_CATALOG_CONN procedure. This action is only done when you no longer plan on using Data Catalog and the external tables that are derived from the catalog. This action deletes Data Catalog metadata, and drops synced external tables from the Autonomous Database instance. The custom properties on Data Catalog and OCI policies are not affected. **More information**: UNSET_DATA_CATALOG_CONN Procedure

Example: MovieStream Scenario

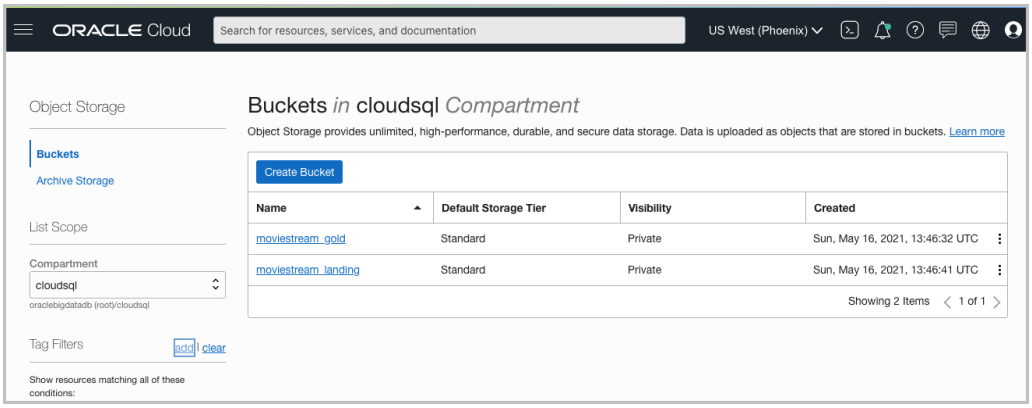
In this scenario, Moviestream is capturing data in a landing zone on object storage. Much of this data, but not necessarily all, is then used to feed an Autonomous Database. Prior to feeding Autonomous Database, the data is transformed, cleansed and subsequently stored in the "gold" area.

Data Catalog is used to harvest these sources and then provide a business context to the data. Data Catalog metadata is shared with Autonomous Database, allowing Autonomous Database users to query those data sources using Oracle SQL. This data may be loaded into Autonomous Database or queried dynamically using external tables.

For more information on using Data Catalog, see Data Catalog Documentation.

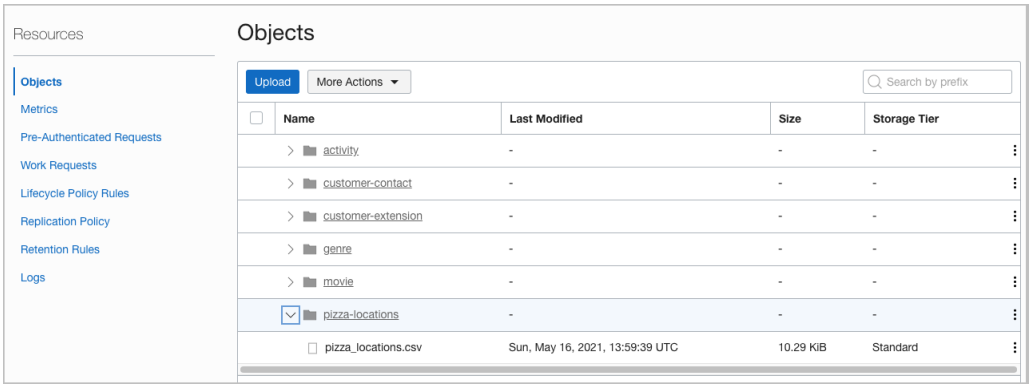
1. Object Store - Review buckets, folders and files
   a. Review the buckets in your object store.

      For example, below are the landing (`moviestream_landing`) and gold zone (`moviestream_gold`) buckets in object storage:



   b. Review the folders and files in the object store buckets.

For example, below are the folders in the landing bucket (`moviestream_landing`) in object storage:

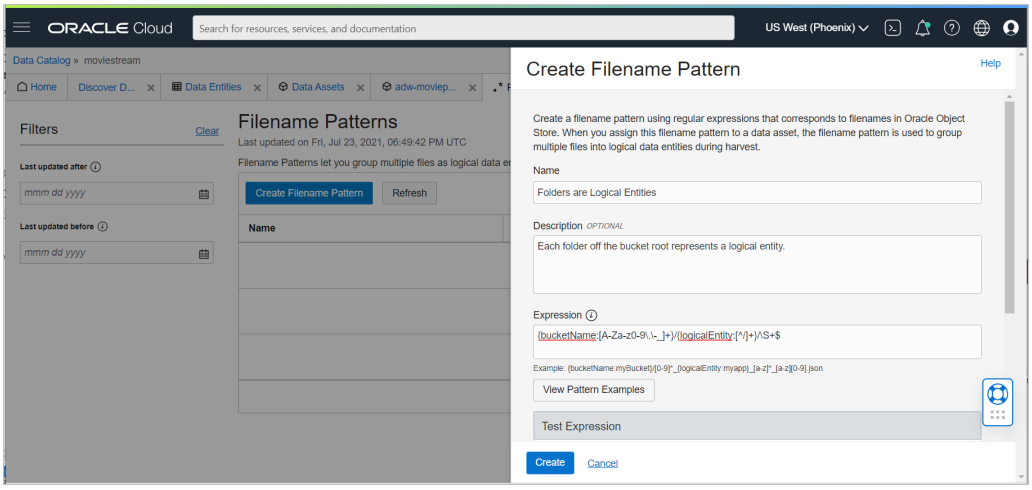


2. Data Catalog - Create filename patterns

   a. Inform Data Catalog how your data is organized using filename patterns. These are regular expressions used to categorize files. The filename patterns are used by the Data Catalog harvester to derive logical entities. The following two filename patterns are used to harvest the buckets in the MovieStream example. See Harvesting Object Storage Files as Logical Data Entities for further details on creating filename patterns.

Hive-style	Folder-style
`{bucketName:.*}/{logicalEntity:[^/]+}.db/{logicalEntity:[^/]+}/.*`	`{bucketName:[\w]+}/{logicalEntity:[^/]+}(?<!.db)/.*$`
• Creates logical entities for sources that contain ".db" as the first part of the object name. • To ensure uniqueness within the bucket, the resulting name is (db-name).(folder name)	• Creates a logical entity based on the folder name off of the root • To prevent duplication with Hive, object names that have ".db" in them are skipped.

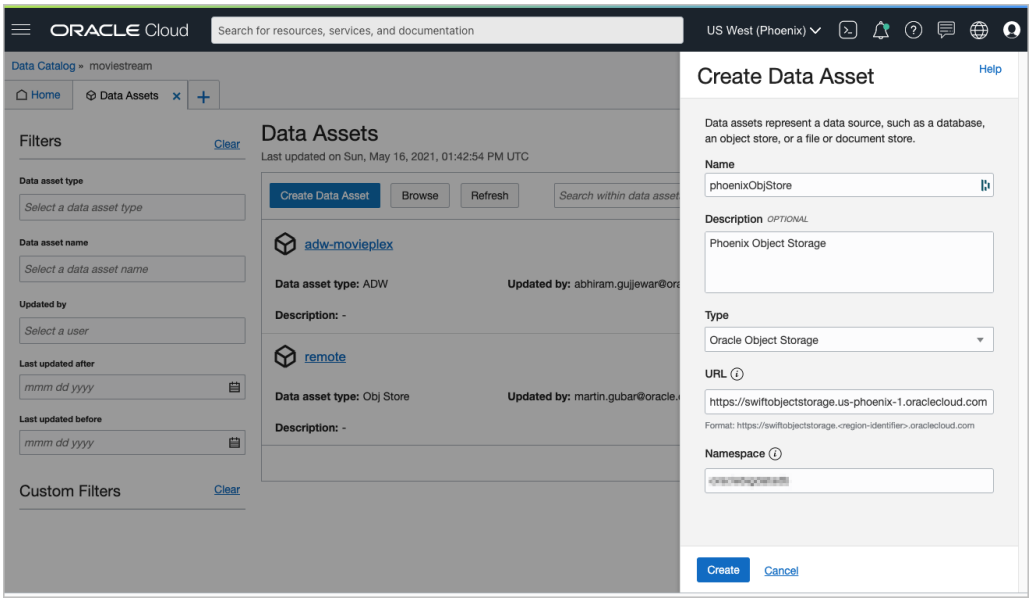
   b. To create filename patterns, go to the **Filename Patterns** tab for your Data Catalog and click **Create Filename Pattern**. For example, the following is the **Create Filename Pattern** tab for the `moviestream` Data Catalog:



3. Data Catalog - Data Asset Creation

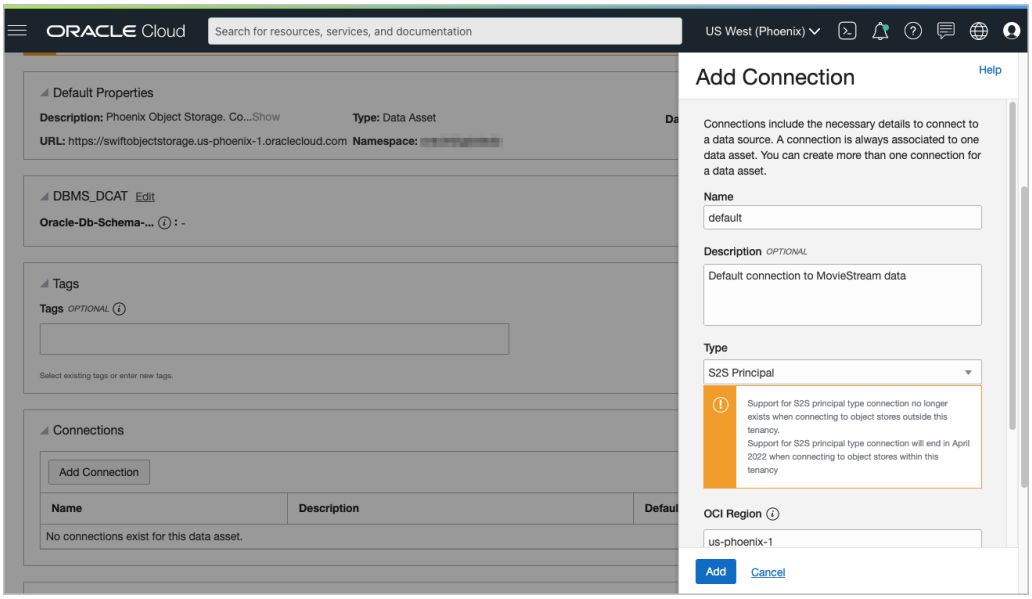
   a. Create a data asset that is used to harvest data from your object store.

For example, a data asset named `phoenixObjStore` is created in the `moviestream` Data Catalog:



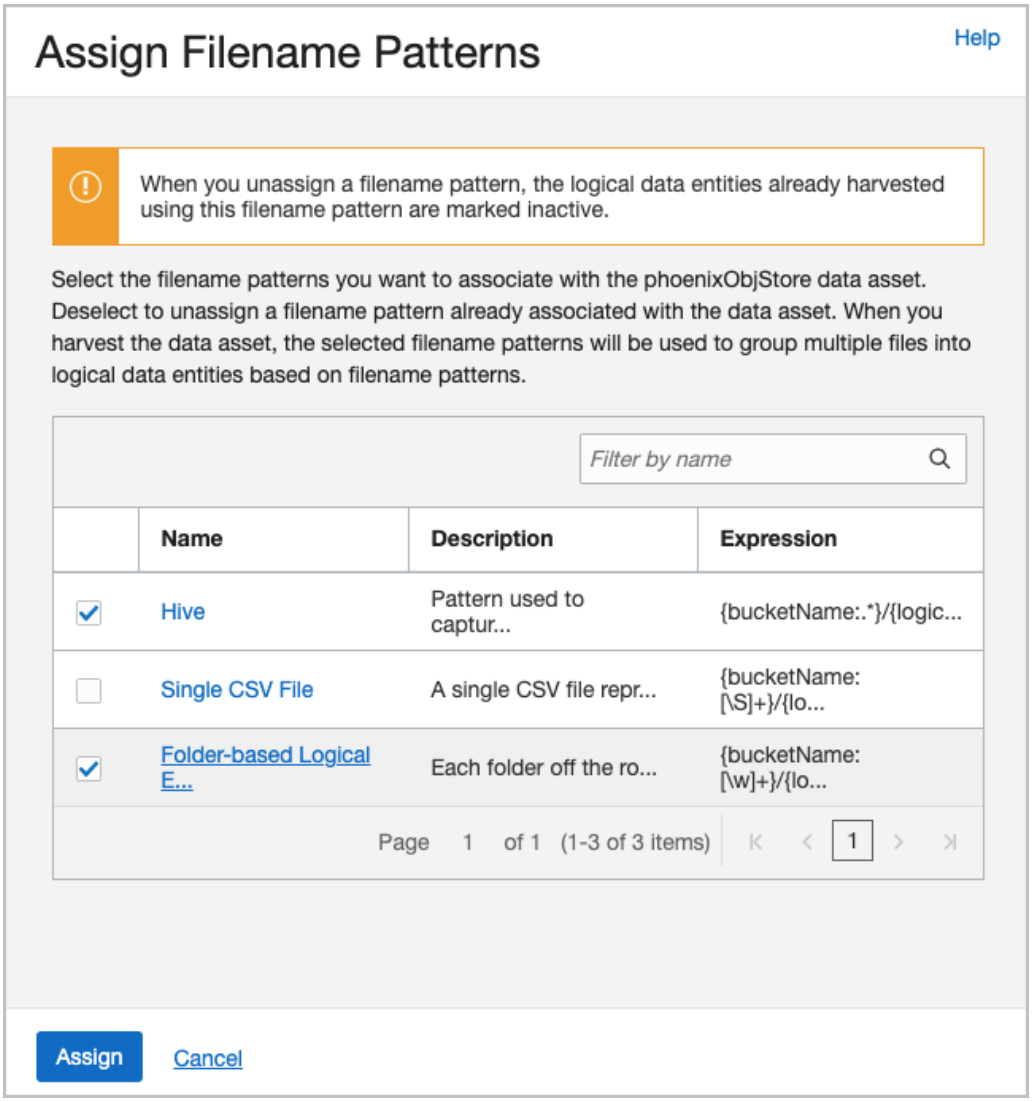
b. Add a connection to your data asset.

In this example, the data asset connects to the compartment for the `moviestream` object storage resource.



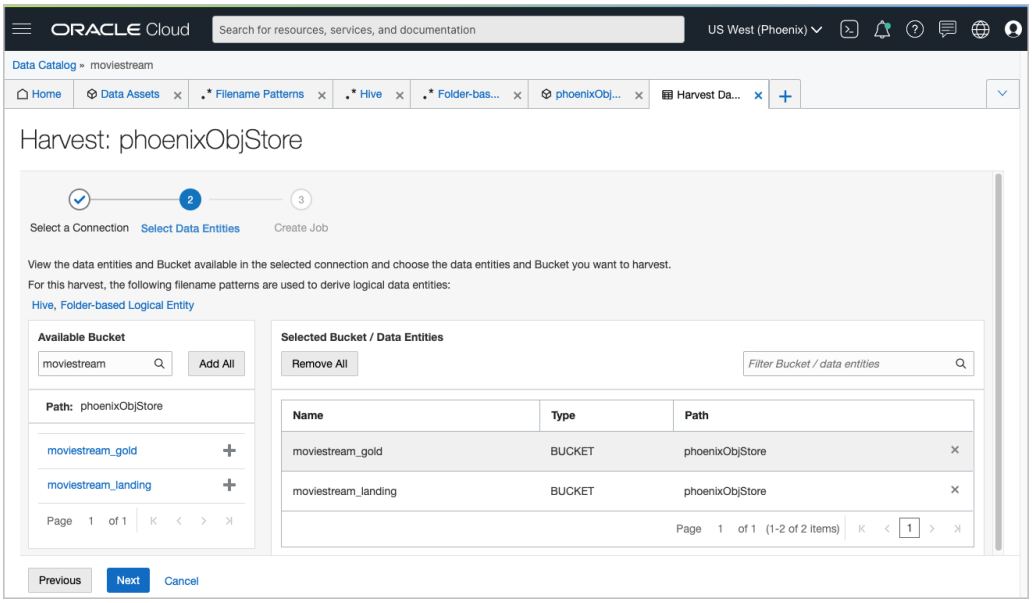
c. Now, associate your filename patterns with your data asset. Select **Assign Filename Patterns**, check the patterns you want and click **Assign**.

For example, here are the patterns assigned to the `phoenixObjStore` data asset:



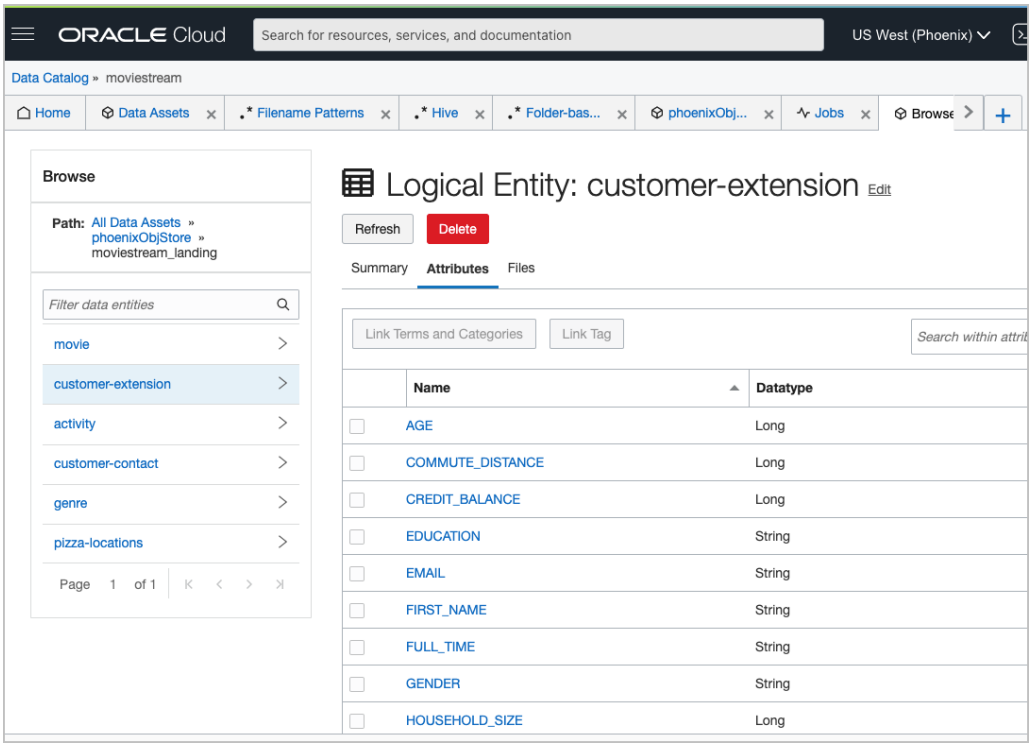
4. Data Catalog - Harvest data from object store
   a. Harvest the Data Catalog data asset. Select the object store buckets containing the source data.

In this example, the `moviestream_gold` and `moviestream_landing` buckets from object store are selected for harvesting.



b. After running the job, you see the logical entities. Use the **Browse Data Assets** to review them.

In this example, you are looking at the `customer-extension` logical entity and its attributes.



If you have a glossary, Data Catalog recommends categories and terms to associate with the entity and its attributes. This provides a business context for the items. Schemas, tables and columns are oftentimes not self-explanatory.

In our example, we want to differentiate between the different types of buckets and the meaning of their content:

- what is a landing zone?

- how accurate is the data?
- when was it last updated?
- what is the definition of a logical entity or its attribute

5. Autonomous Database - Connect to Data Catalog

   Connect Autonomous Database to Data Catalog. You need to ensure that the credential used to make that connection is using an OCI principal that is authorized to access the Data Catalog asset. For further information, see Data Catalog Policies and Access Cloud Resources by Configuring Policies and Roles.

   a. Connect to Data Catalog

```
-- Variables are used to simplify usage later
define oci_credential = 'OCI$RESOURCE_PRINCIPAL'
define dcat_ocid = 'ocid1.datacatalog.oc1.iad.aaaaaaaardp66bg....twiq'
define dcat_region='us-ashburn-1'
define uri_root =    'https://objectstorage.us-
ashburn-1.oraclecloud.com/n/mytenancy/b/landing/o'
define uri_private = 'https://objectstorage.us-
ashburn-1.oraclecloud.com/n/mytenancy/b/private_data/o'

-- Run as admin
-------
-- Enable resource principal support
-------
exec dbms_cloud_admin.enable_resource_principal();

-- Test to make sure credential was created. Returns a row if it was
successful
select *
from dba_credentials
where credential_name = 'OCI$RESOURCE_PRINCIPAL' and owner = 'ADMIN';

-- Query a private bucket to test the principal and privileges.
select *
from dbms_cloud.list_objects('&oci_credential', '&uri_private/');

--------
-- Set the credentials to use for object store and data catalog
-- Connect to Data Catalog
-- Review connection
---------
-- Set credentials
exec dbms_dcat.set_data_catalog_credential(credential_name =>
'&oci_credential');
exec dbms_dcat.set_object_store_credential(credential_name =>
'&oci_credential');

-- Connect to Data Catalog
begin
   dbms_dcat.set_data_catalog_conn (
     region => '&dcat_region',
     catalog_id => '&dcat_ocid');
end;
/
```

```
-- Review the connection
select * from all_dcat_connections;
```

b. Sync Data Catalog with Autonomous Database. Here, we'll sync all the object storage assets:

```
-- Sync Data Catalog with Autonomous Database
---- Let's sync all of the assets.
begin
 dbms_dcat.run_sync('{"asset_list":["*"]}');
end;
/

-- View log
select type, start_time, status, logfile_table from
user_load_operations;  -- Logfile_Table will have the name of the table
containing the full log.
select * from dbms_dcat$1_log;

-- View the new external tables
select * from dcat_entities;
select * from dcat_attributes;
```

c. Autonomous Database - Now start running queries against object store.

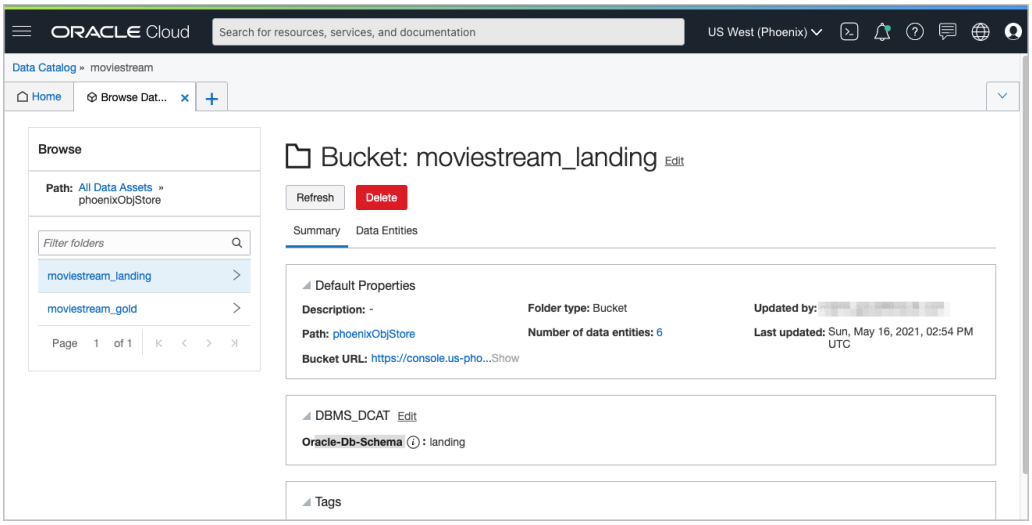
```
-- Query the Data !
select *from dcat$phoenixobjstore_moviestream_gold.genre
;
```

6. Change schemas for objects

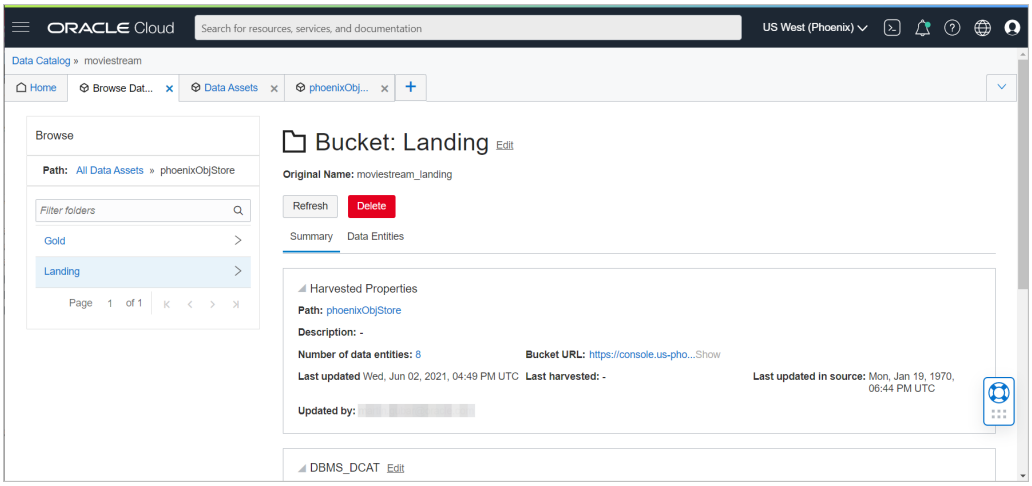
   The default schema names are rather complicated. Let's simplify them by specifying both the asset and the folder's `Oracle-Db-Schema` custom attribute in Data Catalog. Change the data asset to `PHX` and the folders to `landing` and `gold` respectively. The schema is a concatenation of the two.

   a. From Data Catalog, navigate to the `moviestream_landing` bucket and change the asset to `landing` and `gold` respectively.

      Before change:



After change:



b. Run another sync.

Example: Partitioned Data Scenario

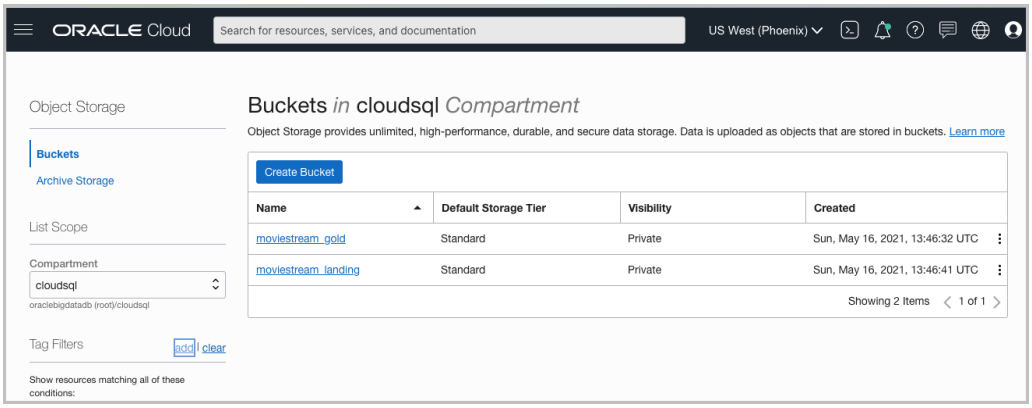
This scenario illustrates how to create external tables in Autonomous Database that are based on Data Catalog logical entities harvested from partitioned data in Object Store.

The following example is based on Example: MovieStream Scenario and has been adapted to demonstrate integrating with partitioned data. Data Catalog is used to harvest these sources and then provide a business context to the data. For further details about this example, see Example: MovieStream Scenario .

For more information on using Data Catalog, see Data Catalog Documentation.

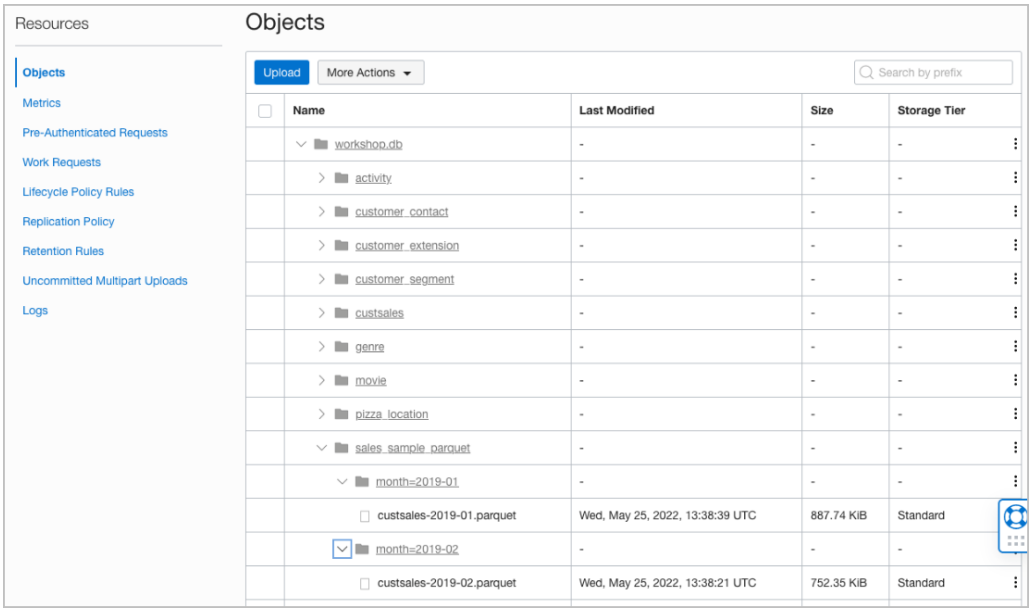
1. Object Store - Review buckets, folders and files
   a. Review the buckets in your object store.

For example, below are the landing (`moviestream_landing`) and gold zone (`moviestream_gold`) buckets in object storage:



b. Review the folders and files in the object store buckets.

For example, below are the folders in the landing bucket (`moviestream_landing`) in object storage:

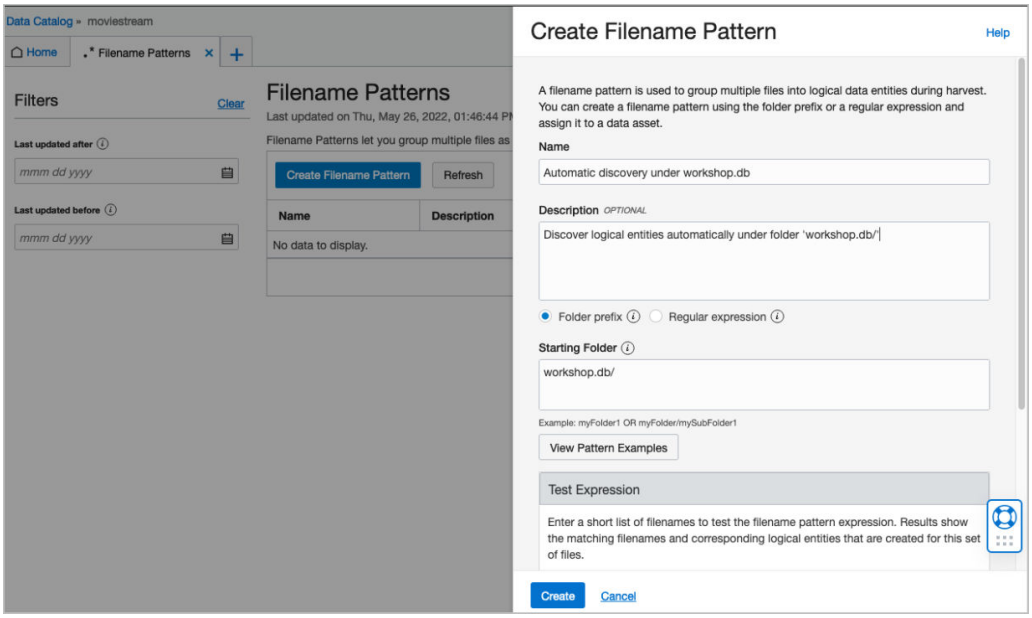


2. Data Catalog - Create filename patterns

a. Inform Data Catalog how your data is organized using filename patterns. These are folder prefixes or regular expressions used to categorize files. The filename patterns are used by the Data Catalog harvester to derive logical entities. When a folder prefix is specified, the Data Catalog automatically generates logical entities from the specified folder prefix in the object store. The following filename pattern is used to harvest the buckets in the MovieStream example. See Harvesting Object Storage Files as Logical Data Entities for further details on creating filename patterns.

Folder prefix	Description
`workshop.db/`	Creates logical entities for sources that contain "workshop.db" path in the object store.

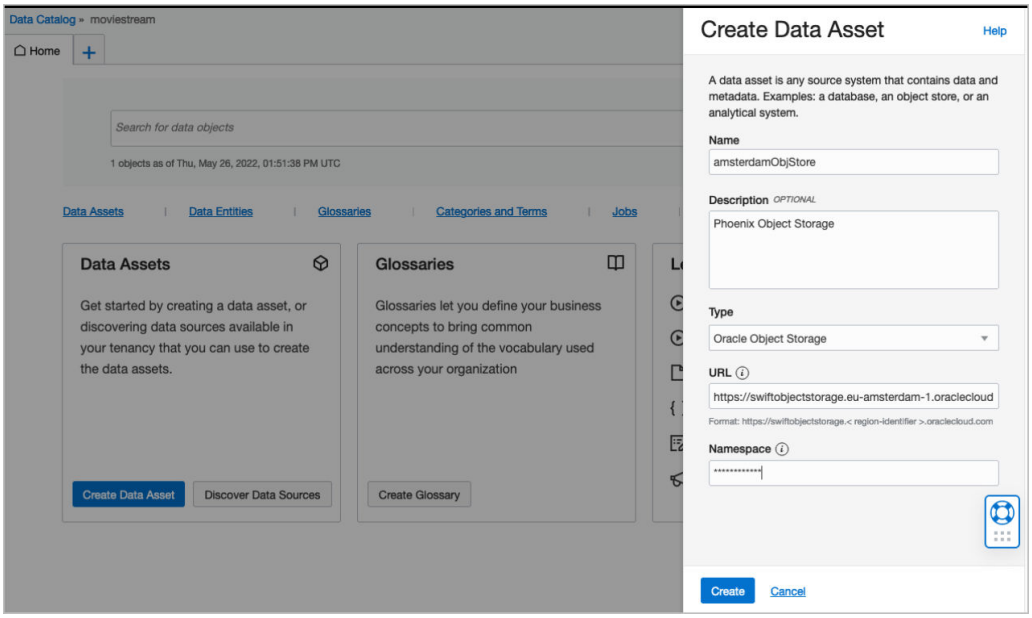
b. To create filename patterns, go to the **Filename Patterns** tab for your Data Catalog and click **Create Filename Pattern**. For example, the following is the **Create Filename Pattern** tab for the `moviestream` Data Catalog:



3. Data Catalog - Data Asset Creation

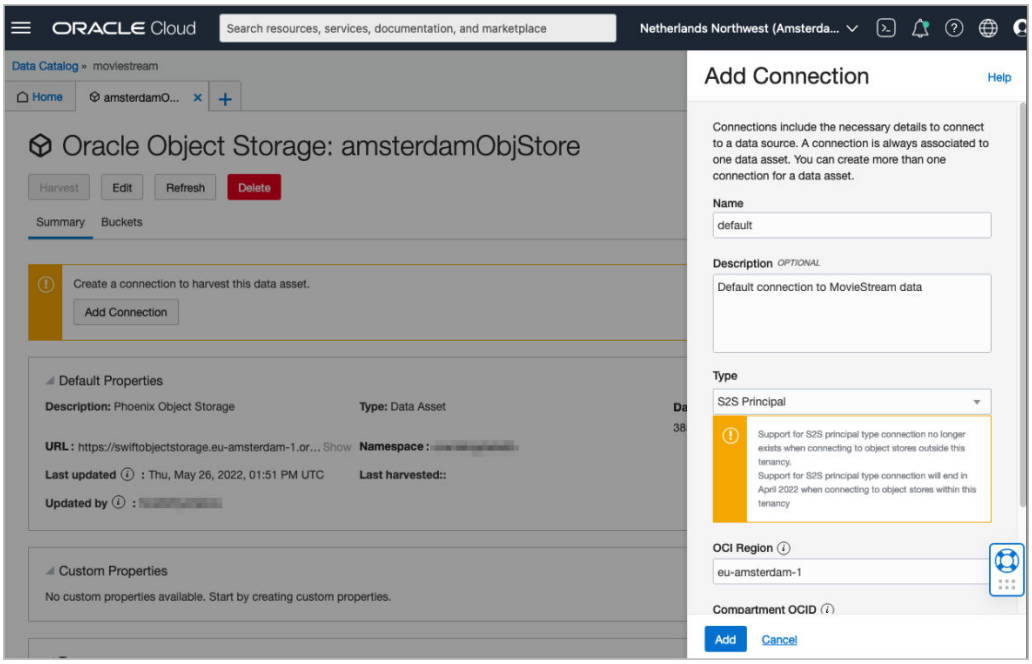
   a. Create a data asset that is used to harvest data from your object store.

      For example, a data asset named `amsterdamObjStore` is created in the `moviestream` Data Catalog:



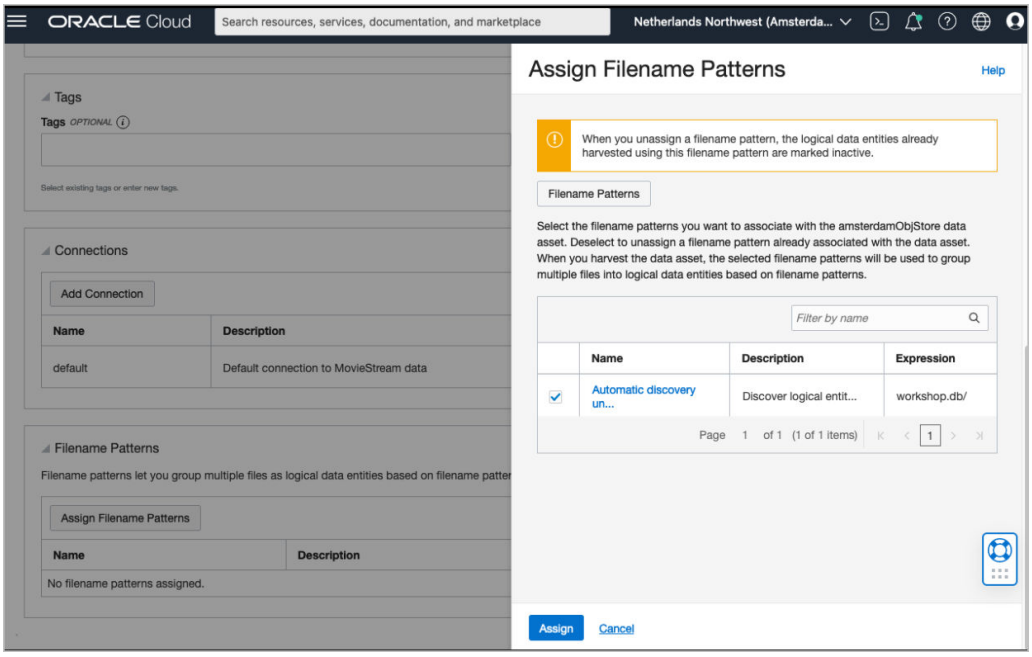
   b. Add a connection to your data asset.

In this example, the data asset connects to the compartment for the `moviestream` object storage resource.



c. Now, associate your filename patterns with your data asset. Select **Assign Filename Patterns**, check the patterns you want and click **Assign**.

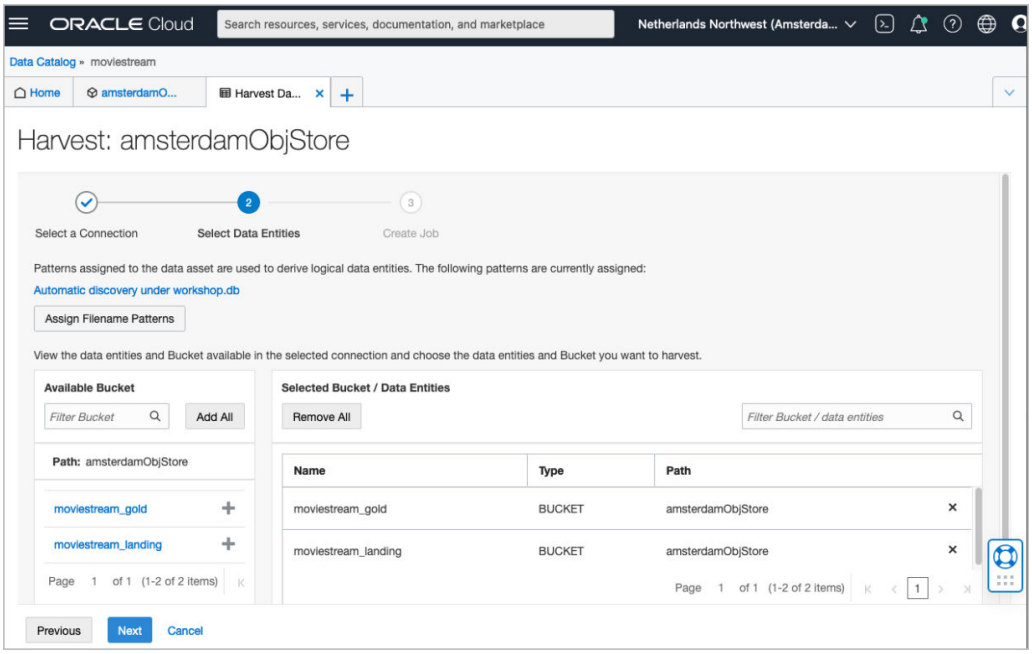
For example, here are the patterns assigned to the `amsterdamObjStore` data asset:



4. Data Catalog - Harvest data from object store

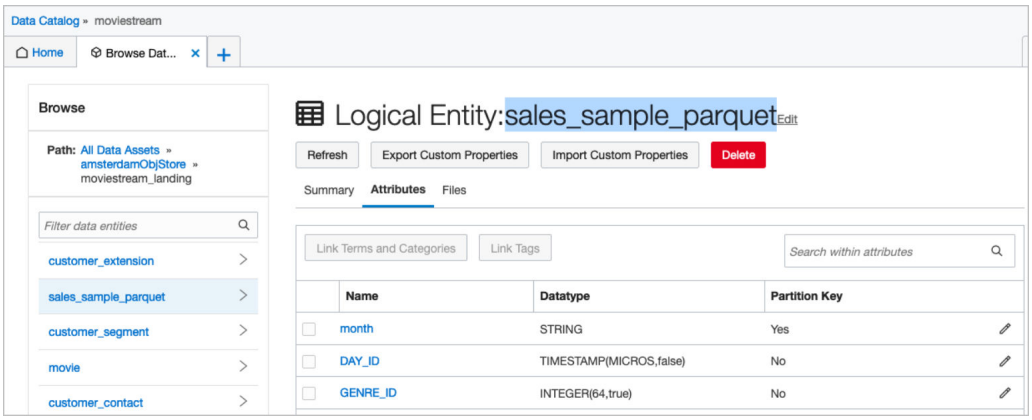
   a. Harvest the Data Catalog data asset. Select the object store buckets containing the source data.

In this example, the `moviestream_gold` and `moviestream_landing` buckets from object store are selected for harvesting.



b. After running the job, you see the logical entities. Use the **Browse Data Assets** to review them.

In this example, you are looking at the `sales_sample_parquet` logical entity and its attributes. Note that Data Catalog has identified the `month` attribute as partitioned.



5. Autonomous Database - Connect to Data Catalog

Connect Autonomous Database to Data Catalog. You need to ensure that the credential used to make that connection is using an OCI principal that is authorized to access the Data Catalog asset. For further information, see Data Catalog Policies and Access Cloud Resources by Configuring Policies and Roles.

a. Connect to Data Catalog

```
-- Variables are used to simplify usage later
define oci_credential = 'OCI$RESOURCE_PRINCIPAL'
define dcat_ocid = 'ocid1.datacatalog.oc1.eu-
amsterdam-1....leguurn3dmqa'
define dcat_region='eu-amsterdam-1'
define uri_root =    'https://objectstorage.us-
ashburn-1.oraclecloud.com/n/mytenancy/b/landing/o'
```

```
define uri_private = 'https://objectstorage.us-
ashburn-1.oraclecloud.com/n/mytenancy/b/private_data/o'

-- Run as admin
-------
-- Enable resource principal support
-------
exec dbms_cloud_admin.enable_resource_principal();

-- Test to make sure credential was created. Returns a row if it was
successful
select *
from dba_credentials
where credential_name = 'OCI$RESOURCE_PRINCIPAL' and owner = 'ADMIN';

-- Query a private bucket to test the principal and privileges.
select *
from dbms_cloud.list_objects('&oci_credential', '&uri_private/');

--------
-- Set the credentials to use for object store and data catalog
-- Connect to Data Catalog
-- Review connection
---------
-- Set credentials
exec dbms_dcat.set_data_catalog_credential(credential_name =>
'&oci_credential');
exec dbms_dcat.set_object_store_credential(credential_name =>
'&oci_credential');

-- Connect to Data Catalog
begin
   dbms_dcat.set_data_catalog_conn (
     region => '&dcat_region',
     catalog_id => '&dcat_ocid');
end;
/
-- Review the connection
select * from all_dcat_connections;
```

b. Sync Data Catalog with Autonomous Database. Here, we'll sync all the object storage assets:

```
-- Sync Data Catalog with Autonomous Database
---- Let's sync all of the assets.
begin
 dbms_dcat.run_sync('{"asset_list":["*"]}');
end;
/

-- View log
select type, start_time, status, logfile_table from
user_load_operations;  -- Logfile_Table will have the name of the table
containing the full log.
select * from dbms_dcat$1_log;
```

```
-- View the new external tables
select * from dcat_entities;
select * from dcat_attributes;
```

c. Autonomous Database - Now start running queries against object store.

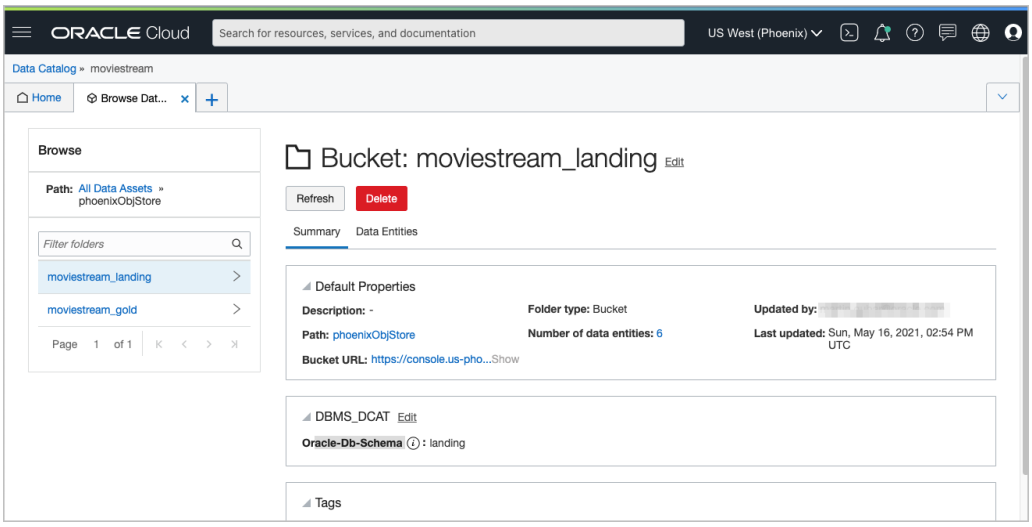
```
-- Query the Data !
select count(*) from
DCAT$AMSTERDAMOBJSTORE_MOVIESTREAM_LANDING.SALES_SAMPLE_PARQUET;

-- Examine the generated partitioned table
select
dbms_metadata.get_ddl('TABLE','SALES_SAMPLE_PARQUET','DCAT$AMSTERDAMOBJS
TORE_MOVIESTREAM_LANDING') from dual;

CREATE TABLE
"DCAT$AMSTERDAMOBJSTORE_MOVIESTREAM_LANDING"."SALES_SAMPLE_PARQUET"
(    "MONTH" VARCHAR2(4000) COLLATE "USING_NLS_COMP",
     "DAY_ID" TIMESTAMP (6),
     "GENRE_ID" NUMBER(20,0),
     "MOVIE_ID" NUMBER(20,0),
     "CUST_ID" NUMBER(20,0),
...
)  DEFAULT COLLATION "USING_NLS_COMP"
ORGANIZATION EXTERNAL
 ( TYPE ORACLE_BIGDATA
   ACCESS PARAMETERS
   ( com.oracle.bigdata.fileformat=parquet
     com.oracle.bigdata.filename.columns=["MONTH"]
     com.oracle.bigdata.file_uri_list="https://swiftobjectstorage.eu-
amsterdam-1.oraclecloud.com/v1/tenancy/moviestream_landing/workshop.db/
sales_sample_parquet/*"
...
   )
 )
REJECT LIMIT 0
PARTITION BY LIST ("MONTH")
(PARTITION "P1"  VALUES (('2019-01'))
   LOCATION ( 'https://swiftobjectstorage.eu-
amsterdam-1.oraclecloud.com/v1/tenancy/moviestream_landing/workshop.db/
sales_sample_parquet/month=2019-01/*'),
PARTITION "P2"  VALUES (('2019-02'))
   LOCATION ( 'https://swiftobjectstorage.eu-
amsterdam-1.oraclecloud.com/v1/tenancy/moviestream_landing/workshop.db/
sales_sample_parquet/month=2019-02/*'),
...PARTITION "P24"  VALUES (('2020-12'))
   LOCATION ( 'https://swiftobjectstorage.eu-
amsterdam-1.oraclecloud.com/v1/tenancy/moviestream_landing/workshop.db/
sales_sample_parquet/month=2020-12/*'))
PARALLEL
```

6. Change schemas for objects

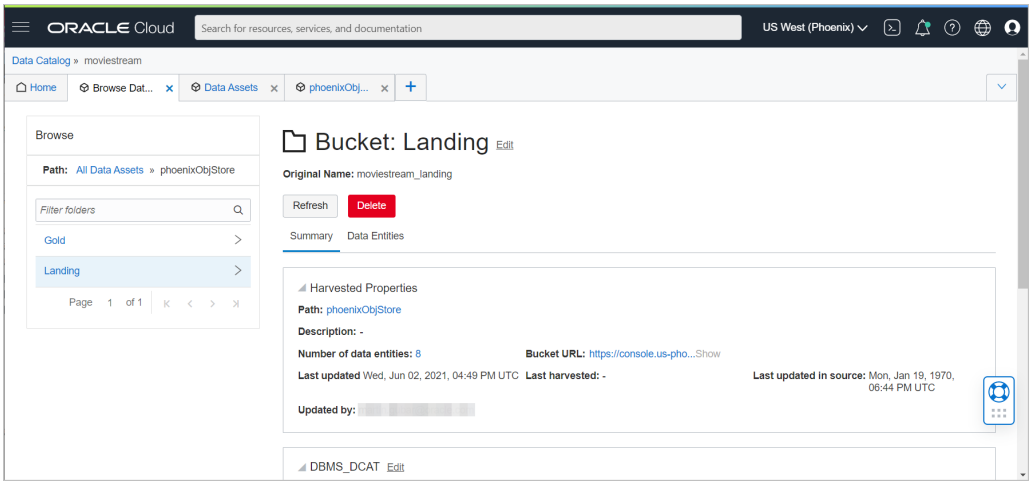
The default schema names are rather complicated. Let's simplify them by specifying both the asset and the folder's `Oracle-Db-Schema` custom attribute in Data Catalog. Change the data asset to `PHX` and the folders to `landing` and `gold` respectively. The schema is a concatenation of the two.

a. From Data Catalog, navigate to the `moviestream_landing` bucket and change the asset to `landing` and `gold` respectively.

Before change:



After change:



b. Run another sync.

Query External Data with AWS Glue Data Catalog

Autonomous Database supports a system for synchronizing with an Amazon AWS Glue Data Catalog instance.

- About Querying with AWS Glue Data Catalog
  Autonomous Database allows you to synchronize with Amazon Web Service (AWS) Glue Data Catalog metadata. A database external table is automatically created by Autonomous Database for every table harvested by AWS Glue about data stored in Amazon Simple Storage Service (S3). Users can query data stored in S3 from Autonomous Database without having to manually derive the schema for the external data sources and create external tables.

- Concepts Related to Querying with AWS Glue Data Catalog
  An understanding of the following concepts is necessary for querying with Amazon Web Service (AWS) Glue data catalogs.
- Mapping between Autonomous Database and AWS Glue
  During the synchronization process, external tables are created in Autonomous Database derived from the AWS Glue Data Catalog databases and tables over Amazon S3.
- User Workflow for Querying with AWS Glue Data Catalog
  The basic user workflow for querying AWS S3 data with AWS Glue Data Catalog involves connecting to AWS Glue Data Catalog, synchronizing with Autonomous Database to automatically create external tables, and then querying the S3 data.
- Example: Query with AWS Glue Data Catalog
  This example steps you through the process of running queries over datasets stored in Amazon Simple Storage Service (Amazon S3) using an AWS Glue Data Catalog.

About Querying with AWS Glue Data Catalog

Autonomous Database allows you to synchronize with Amazon Web Service (AWS) Glue Data Catalog metadata. A database external table is automatically created by Autonomous Database for every table harvested by AWS Glue about data stored in Amazon Simple Storage Service (S3). Users can query data stored in S3 from Autonomous Database without having to manually derive the schema for the external data sources and create external tables.

Amazon AWS Glue Data Catalog is a centralized metadata management service that helps data professionals discover data and supports data governance in AWS cloud. An Autonomous Database instance has the ability to synchronize automatically data catalog metadata with AWS Glue Data Catalog allowing database users to immediately use Autonomous Database to query data stored in the AWS cloud.

Synchronizing with AWS Glue Data Catalog has the same properties as synchronizing with OCI Data Catalog. Synchronization is dynamic, keeping the database up-do-date with respect to changes to the underlying data, reducing administration cost as it automatically maintains hundreds to thousands of tables.

Concepts Related to Querying with AWS Glue Data Catalog

An understanding of the following concepts is necessary for querying with Amazon Web Service (AWS) Glue data catalogs.

AWS Glue Data Catalog: Database
An AWS Glue database represents a collection of relational table definitions, organized in a logical group. Each AWS Glue data catalog instance manages multiple databases.

AWS Glue Data Catalog: Table
An AWS Glue table represents a relational table over data stored in the AWS cloud. An AWS Glue table defines the schema of the underlying data and consists of column information, partition information, serialization information, storage information, statistics, user-defined metadata and other metadata. Tables in AWS Glue data catalog can be created manually, or automatically using an AWS Glue crawler.

Glue Data Catalog: Crawler
You can use a crawler to populate the AWS Glue data catalog with tables. This is the primary method used by most AWS Glue users. A crawler can crawl multiple data stores in a single run. Upon completion, the crawler creates or updates one or more tables in your data catalog. Extract, transform, and load (ETL) jobs that you define in AWS Glue use these data catalog

tables as sources and targets. The ETL job reads from and writes to the data stores that are specified in the source and target data catalog tables.
AWS Glue tables can be created manually by the user or automatically using a predefined or a custom crawler. Crawlers connect to the underlying data stores (for example, Amazon S3), invoke classifiers for deriving the schema of the data, and create AWS Glue tables for storing the inferred metadata. AWS Glue provides classifiers for common file types, such as CSV, JSON, Parquet, and AVRO.

Mapping between Autonomous Database and AWS Glue

During the synchronization process, external tables are created in Autonomous Database derived from the AWS Glue Data Catalog databases and tables over Amazon S3.

AWS Glue organizes collected metadata in databases and tables. An AWS Glue database is a collection of relational table definitions. AWS Glue tables that describe the common schema and properties of the files associated with the table.

AWS Glue follows the relational model for representing attributes. For mapping hierarchical schemas to relational schemas, AWS Glue infers the schema of the semi-structured data and flattens the data to a relational schema using an ETL process.

The following table represents the mapping between OCI Data Catalog concepts and AWS Glue Data Catalog Concepts.

Table 3-4 OCI Data Catalog to AWS Glue Data Catalog Mapping

OCI Data Catalog	AWS Glue Data Catalog	Oracle Database
Data Asset	Database	Schema
Folder	(Bucket)	Schema
Logical Entity	Table	Table

User Workflow for Querying with AWS Glue Data Catalog

The basic user workflow for querying AWS S3 data with AWS Glue Data Catalog involves connecting to AWS Glue Data Catalog, synchronizing with Autonomous Database to automatically create external tables, and then querying the S3 data.

The Database Data Catalog Administrator creates a connection between the Autonomous Database instance and an AWS Glue Data Catalog instance, then configures and runs a synchronization (sync) between the AWS Glue Data Catalog and Autonomous Database. Autonomous Database automatically creates an external table for tables harvested by AWS Glue about data stored in S3.
The Database Data Catalog Query Admin or Database Admin grants READ access to the generated external tables so that Data Analysts and other database users can browse and query Autonomous Database without having to manually derive the schema for the external data sources and create external tables.

Users

The table below describes the different types of users that perform the user workflow actions.

User	Description
Database Data Catalog Administrator	Database user with `DCAT_SYNC` role.
Database Data Catalog Query Administrator	Database user able to grant access on automatically created external tables to other users.

User	Description
Data Analyst	Database user on Autonomous Database querying data in AWS S3 either by querying automatically created external tables or by interacting directly with AWS Glue Data Catalog.
AWS Glue Data Catalog User	AWS user with access to an AWS Glue Data Catalog.
AWS S3 Object Storage User	AWS user with access to data stored in AWS S3

User Workflow

The table below describes each action included in the workflow and what type of user can perform the action.

> **Note:**
>
> The `DBMS_DCAT` package is available for performing the tasks required to query AWS S3 object storage using AWS Glue Data Catalog. See DBMS_DCAT Package.

Action	Who is the user	Description
Create policies	Database Data Catalog Administrator	The Autonomous Database user credential must have the appropriate permissions to access AWS Glue Data Catalog and to read from S3 object storage. **More information**: Required Credentials and IAM Policies.
Create credentials	Database Data Catalog Administrator	Ensure database credentials are in place to access AWS Glue Data Catalog and to query S3 object storage. The user calls `DBMS_CLOUD.CREATE_CREDENTIAL` to create user credentials. **Note:** Only Amazon Web Services (AWS) credentials are supported. AWS Amazon Resource Names (ARN) credentials are not supported. **More information**: DBMS_CLOUD CREATE_CREDENTIAL Procedure

Action	Who is the user	Description
Connect	Database Data Catalog Administrator	Establish a connection between an Autonomous Database instance and an AWS Glue Data Catalog instance. The connection uses the privileges of the AWS Glue Data Catalog User. Connections from an Autonomous Database instance to multiple AWS Glue Data Catalog instances is supported. To initiate a connection between an Autonomous Database instance and an AWS Glue Data Catalog instance the user: 1. Calls `DBMS_DCAT.SET_DATA_CATALOG_CREDENTIAL` using an AWS credential (access key/secret access key) to specify the target AWS Glue Data Catalog instance. 2. Calls the `DBMS_DCAT.SET_OBJECT_STORE_CREDENTIAL` using an AWS credential (access key/secret access key) to access the AWS S3 object storage. 3. Calls `DBMS_DCAT.SET_DATA_CATALOG_CONN` using an AWS Glue service endpoint Once the connection has been made, Autonomous Database stores the associated metadata, such as the AWS Glue catalog id, region, endpoint, and credential objects. **More information**: SET_DATA_CATALOG_CONN Procedure, UNSET_DATA_CATALOG_CONN Procedure, SET_DATA_CATALOG_CREDENTIAL Procedure, SET_OBJECT_STORE_CREDENTIAL Procedure.
Synchronize	Database Data Catalog Administrator	The user can manually start a synchronization with connected AWS Glue Data Catalogs using `DBMS_DCAT.RUN_SYNC` or create automatic synchronizations using `DBMS_DCAT.CREATE_SYNC_JOB`. Synchronization does the following: • Creates external tables in Autonomous Database that are derived from the connected AWS Glue Data Catalog. • Allows users to preview metadata for AWS Glue databases and tables. • Modifies or drops existing synchronized external tables if AWS Glue metadata has changed. **More information**: See RUN_SYNC Procedure, CREATE_SYNC_JOB Procedure, DROP_SYNC_JOB Procedure, Synchronization Mapping
Monitor Synchronization	Database Data Catalog Administrator	The user can view the sync status by querying the `USER_LOAD_OPERATIONS` view. After the sync process has completed, the user can view a log of the sync results, including details about the mappings to external tables. **More information**: Monitoring and Troubleshooting Loads
Grant privileges	Database Data Catalog Query Administrator, Database Administrator	The database Data Catalog Query Administrator or Database Administrator must grant READ privileges on generated external tables to data analyst users. This allows the data analysts to query the generated external tables.

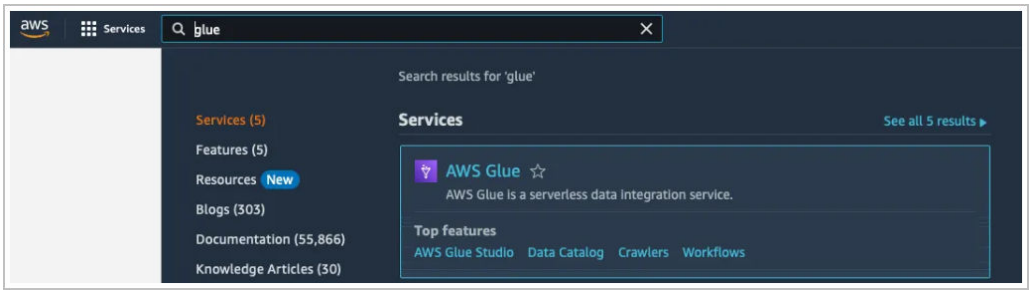
Action	Who is the user	Description
Query	Data Analyst	Data analysts are able to review the synced schemas and tables in the GLUE$* schemas and query the external tables through any tool or application that supports Oracle SQL. Data in S3 is accessed using the privileges of the AWS S3 object storage user. **More information**: Example: Query with AWS Glue Data Catalog
Terminate connections	Database Data Catalog Administrator	To remove an existing Data Catalog association, the user calls the `UNSET_DATA_CATALOG_CONN` procedure. This action is only done when you no longer plan on using the connected AWS Glue Data Catalog and the external tables that are derived from the catalog. This action deletes AWS Glue Data Catalog metadata, and drops synced external tables from the Autonomous Database instance. **More information**: UNSET_DATA_CATALOG_CONN Procedure

Example: Query with AWS Glue Data Catalog

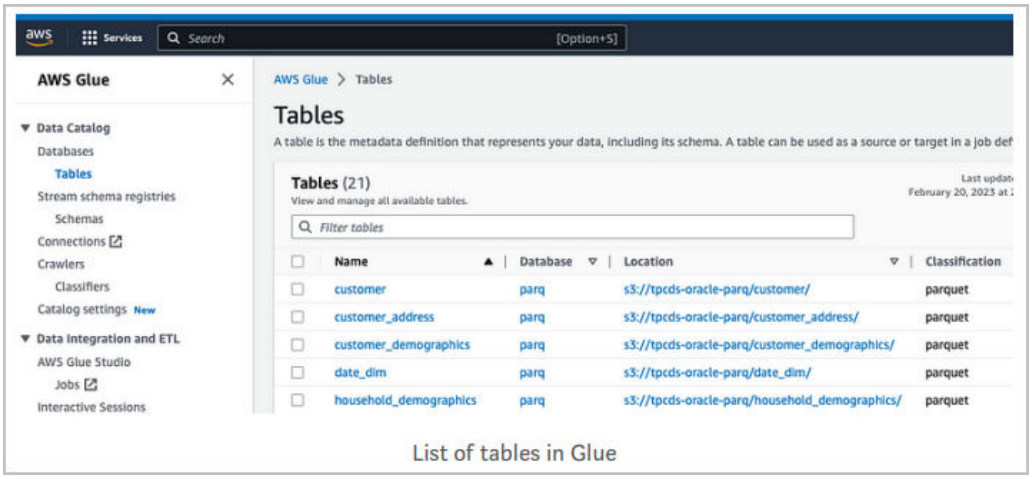
This example steps you through the process of running queries over datasets stored in Amazon Simple Storage Service (Amazon S3) using an AWS Glue Data Catalog.

In this example, metadata in an AWS Glue Data Catalog is inspected to see what Amazon S3 objects have been previously crawled and exist in the data catalog. Autonomous Database is then associated with the AWS Glue Data Catalog and Amazon S3. The data catalog is synchronized with Autonomous Database to create external tables over the datasets stored in Amazon S3. The external tables are used to query the datasets in Amazon S3.

1. Inspect metadata in AWS Glue Data Catalog.
   a. Launch the AWS Glue console.



   b. Navigate to the data catalog, databases and tables to find existing objects.
   In this example, some objects exist in Amazon S3 that AWS Glue has previously crawled and created tables for as shown below:



List of tables in Glue

2. Associate AWS Glue with Autonomous Database.

   a. Create credentials in Autonomous Database.
   The following procedure call includes the access ID and secret key to provide Autonomous Database with access to the underlying data in Amazon S3.

   ```
   exec dbms_cloud.create_credential('CRED_AWS','<access id>', '<access
   key>');
   ```

   b. Associate the credentials with the AWS Glue Data Catalog and Amazon S3 object storage.
   These procedure calls associate the data catalog and object storage, respectively, with the credentials.

   ```
   exec dbms_dcat.set_data_catalog_credential('CRED_AWS');
   exec dbms_dcat.set_object_store_credential('CRED_AWS');
   ```

   c. Set up an AWS region where Glue is running.

   ```
   exec dbms_dcat.set_data_catalog_conn(region => 'us-west-2',
   catalog_type=>'AWS_GLUE');
   ```

3. Synchronize metadata to create external tables in Autonomous Database derived from AWS Glue databases and tables.

   a. Now that the association is done, use the `all_glue_databases` view to find what databases are inside an AWS Glue Data Catalog.

   ```
   select * from all_glue_databases order by name;
   ```

   b. Use the `all_glue_tables` view to get a list of tables available for sync.

   ```
   select * from all_glue_tables order by database_name, name;
   ```

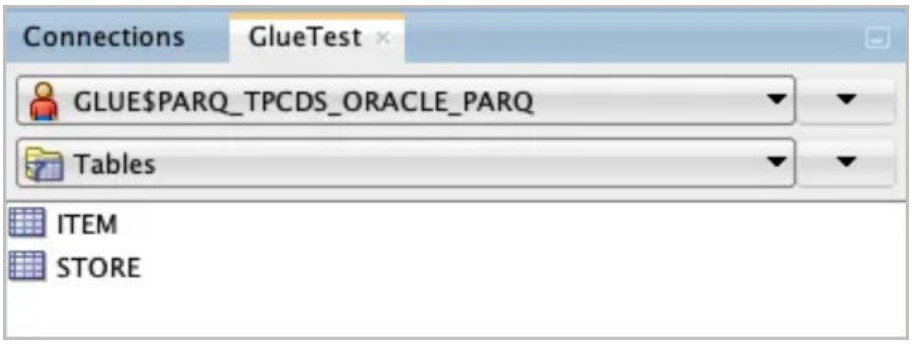
CATALO...	DATA...	NAME	TABLE_TYPE	CLASSIFICATION	DESCRIPTION	OWNER	CREATED_BY
221152...	parq	customer	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	customer_address	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	customer_demographics	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	date_dim	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	household_demographics	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	income_band	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	inventory	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	item	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	promotion	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	reason	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	ship_mode	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	store	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	store_returns	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	store_sales	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	test_glue	EXTERNAL_TABLE	(null)	(null)	221152299178	arn:aws:iam::22115...
221152...	parq	time_dim	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	warehouse	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...

c. Synchronize Autonomous Database with two tables, `store` and `item`, found in the `parq` database.

```
begin
dbms_dcat.run_sync(
    synced_objects => '
        {
            "database_list": [
                {
                "database": "parq",
                "table_list": ["store","item"]
                }
             ]
         }',
error_semantics => 'STOP_ON_ERROR');
end;
/
```

4. Inspect new objects in Autonomous Database and run a query on top of S3.

   a. Use SQL Developer to view new objects created by the previous sync operation. The `GLUE$PARQ_TPCDS_ORACLE_PARQ` schema was generated and named automatically by the `dbms_dcat.run_sync` procedure call.



   b. Run a SQL query over datasets stores in Amazon S3.

```
SELECT * FROM glue$parq_tpcds_oracle_parq.store;
```

Query Apache Iceberg Tables

Autonomous Database supports querying Apache Iceberg tables.

- About Querying Apache Iceberg Tables
  Autonomous Database supports querying of Apache Iceberg tables stored in Amazon Web Services (AWS) or in Oracle Cloud Infrastructure (OCI) Object Storage.
- Concepts Related to Querying Apache Iceberg Tables
  An understanding of the following concepts is helpful for querying Apache Iceberg tables.
- Examples: Querying Apache Iceberg Tables
  These examples show how to query Apache Iceberg tables on Amazon Web Services (AWS) and Oracle Cloud Infrastructure (OCI), using a data catalog and using direct URLs for the root manifest file.

About Querying Apache Iceberg Tables

Autonomous Database supports querying of Apache Iceberg tables stored in Amazon Web Services (AWS) or in Oracle Cloud Infrastructure (OCI) Object Storage.

Supported Configurations

These specific configurations are supported:

- Iceberg tables on AWS:
  - Iceberg tables registered with AWS Glue Data Catalog, created with Spark or Athena. For more information see Use the AWS Glue connector to read and write Apache Iceberg tables with ACID transactions and perform time travel and Using Iceberg tables.
  - Iceberg tables stored on AWS S3 by directly providing the URL for the root metadata file.
- Iceberg tables on OCI:
  - Iceberg tables generated with OCI Data Flow using a Hadoop Catalog.
    For more information, see Oracle Data Flow Examples and Using a Hadoop Catalog.
  - Iceberg tables stored on OCI Object Storage by directly providing the URL for the root metadata file.

Restrictions

- Partitioned Iceberg tables
  Oracle does not support Iceberg partitioned tables.
- Row-level updates of Iceberg tables
  Oracle does not support merge-on-read for Iceberg table updates. Queries encountering deleted files in the Iceberg metadata will fail. For more information about merge-on-read, see Enum RowLevelOperationMode.
- Schema evolution
  The schema for Oracle external tables is fixed and reflects the Iceberg schema version at external table creation time. Queries fail if the Iceberg metadata points to a different schema version compared to the one used to create the Iceberg table. If the Iceberg schema changes after the external table has been created, it is recommended that the external table be recreated.

Concepts Related to Querying Apache Iceberg Tables

An understanding of the following concepts is helpful for querying Apache Iceberg tables.

Iceberg Catalog

The Iceberg catalog is a service that manages table metadata, such table snapshots, the table schema and partitioning information. In order to query the latest snapshot for an Iceberg table, query engines must first access the catalog and obtain the location of the most recent metadata file. There are already a number of available catalog implementations, including AWS Glue, Hive, Nessie, and Hadoop. Autonomous Database supports the AWS Glue catalog and the HadoopCatalog implementation used by Spark.

For more information, see Iceberg Catalog, Optimistic Concurrency, Catalog Implementations, and Class HadoopCatalog.

Metadata Files

The metadata file is a JSON document that keeps track of the table snapshots, partitioning scheme and schema information. The metadata file is the entry point to a hierarchy of manifest lists and manifest files. The manifests track the table's data files along with information including partitioning and column statistics. See the Iceberg Table Specification, for more information.

Transactions

Iceberg supports row-level updates to tables using either copy-on-write or merge-on-read. Copy-on-write generates new data files that reflect the updated rows, while merge-on-read generates new "delete files" that must be merged with the data files during reading. Oracle supports copy-on-write. Queries on iceberg tables fail if they encounter a delete file. For more information, see RowLevelOperationMode.

Schema Evolution

Iceberg supports schema evolution. Schema changes are reflected in the Iceberg metadata using a schema ID. Note that Oracle external tables have a fixed schema, determined by the most current schema version at table creation time. Iceberg queries fail when the queried metadata points to a different schema version compared to the one used at table creation time. For more information, see Schema Evolution.

Partitioning

Iceberg supports advanced partitioning options such as hidden partitioning and partition evolution that rely on processing/altering the table's metadata without costly data layout changes.

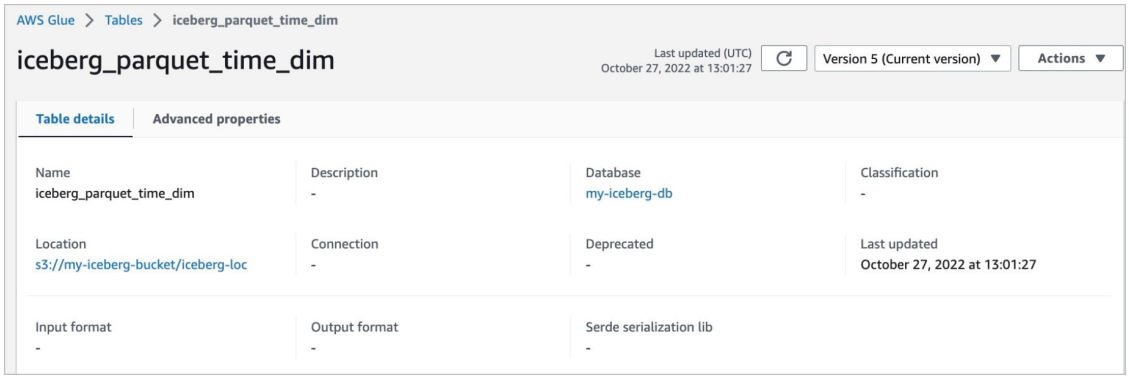
Examples: Querying Apache Iceberg Tables

These examples show how to query Apache Iceberg tables on Amazon Web Services (AWS) and Oracle Cloud Infrastructure (OCI), using a data catalog and using direct URLs for the root manifest file.

For detailed information on creating external tables for Apache Iceberg, see CREATE_EXTERNAL_TABLE Procedure for Apache Iceberg.

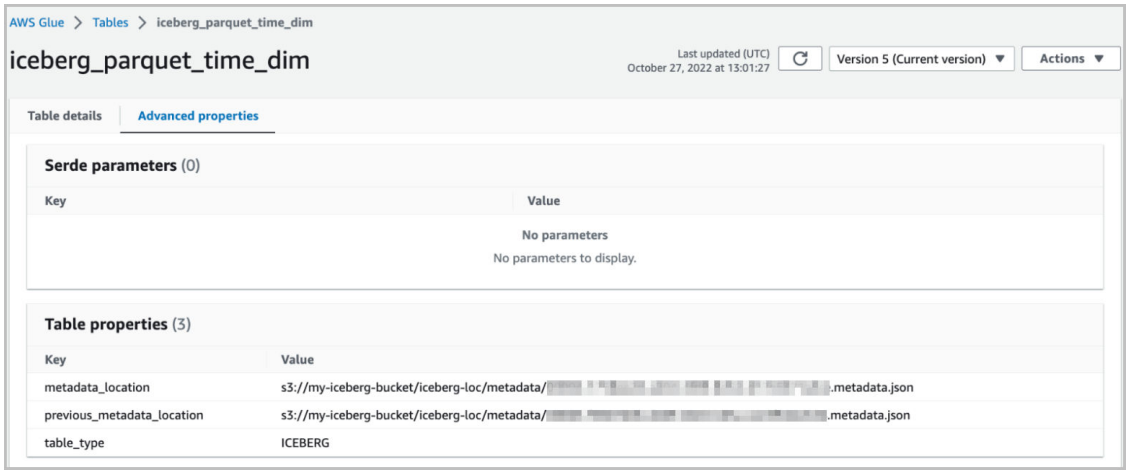
Query an Iceberg table on AWS using a Glue Data Catalog

In this example, we query the Iceberg table `iceberg_parquet_time_dim`.



The table belongs to Glue database `my-iceberg-db` and is stored in folder `s3://my-iceberg-bucket/iceberg-loc`.

The table details for `iceberg_parquet_time_dim` are shown here:



We can create an external table for `iceberg_parquet_time_dim` as follows:

```
BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_TABLE (
    table_name       => 'iceberg_parquet_time_dim',
    credential_name  => 'AWS_CRED',
    file_uri_list    => '',
    format           =>
      '{"access_protocol":
        {
         "protocol_type": "iceberg",
         "protocol_config":
          {
           "iceberg_catalog_type": "aws_glue",
           "iceberg_glue_region": "us-west-1",
           "iceberg_table_path": "my-iceberg-db.iceberg_parquet_time_dim"
          }
        }
      }'
  );
```

```
END;
/
```

In the protocol_config section we specify that the table uses AWS Glue as the catalog type, and set the catalog's region to `us-west-1`.

The credential `AWS_CRED` is created using `dbms_cloud.create_credential` with an AWS API key. The Glue Data Catalog instance is determined by the account ID associated with `AWS_CRED` for region `us-west-1`, as there is a single Glue Data Catalog region for each account. The `iceberg_table_path` element in the `protocol_config` section uses a `$database_name.$table_name` path to specify the Glue table name and database name. Finally, the `column_list` and `field_list` parameters are left null because the table's schema is automatically derived from the Iceberg metadata.

For further information on creating the credential, see CREATE_CREDENTIAL Procedure. For information on AWS Glue resources, see Specifying AWS Glue resource ARNs.

Query an Iceberg table on AWS using the location of the root metadata file

If we know the location of the metadata file for an Iceberg table, we can create an external table without specifying a catalog, as follows:

```
BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_TABLE (
    table_name       => 'iceberg_parquet_time_dim',
    credential_name  => 'AWS_CRED',
    file_uri_list    => 'https://my-iceberg-bucket.s3.us-west-1.amazonaws.com/
iceberg-loc/metadata/00004-1758ee2d-a204-4fd9-8d52-
d17e5371a5ce.metadata.json',
    format           =>'{"access_protocol":{"protocol_type":"iceberg"}}');
END;
/
```

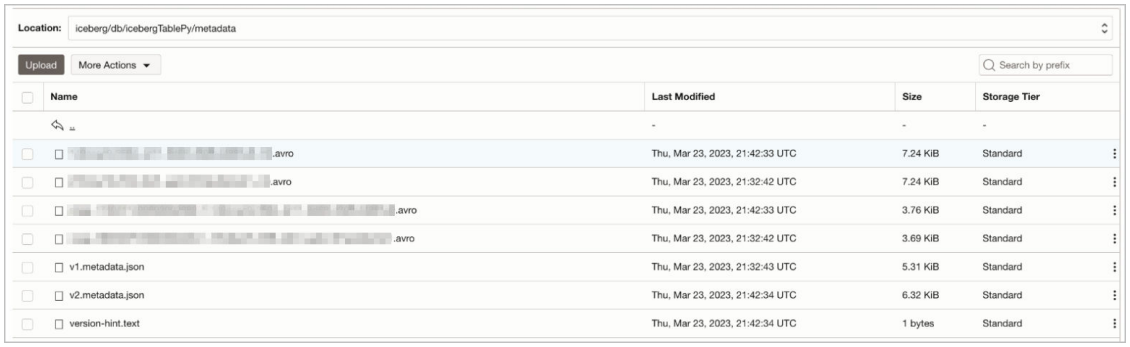
We use the `file_uri_list` parameter to specify the location of the metadata file in AWS S3 virtual-hosted-style URL format. For information on this format, see Methods for accessing an AWS S3 bucket.

In this example, the database accesses the metadata file directly, so there is no need to provide a `protocol_config` section in the `format` parameter. When using the metadata file location to create an external table, the database queries the most recent snapshot referenced by the metadata file. Subsequent updates to the Iceberg table, that create new snapshots and new metadata files, will not be visible to the database.

Query an Iceberg table that uses Hadoop Catalog on OCI

In this example, we query the Iceberg table `icebergTablePy`, created using OCI Data Flow, where Spark uses the HadoopCatalog implementation for the Iceberg catalog. HadoopCatalog uses a `warehouse` directory and puts the Iceberg metadata in a `$database_name/$table_name` subfolder under this directory. It also uses a `version-hint.text` file that contains the version number for the most recent metadata file version. See Iceberg Support on OCI Data Flow for the example on Github.

The sample table `db.icebergTablePy` was created using a `warehouse` folder, named `iceberg`, in OCI bucket `my-iceberg-bucket`. The storage layout on OCI for table `icebergTablePy` is shown below:



Create an external table for table `db.icebergTablePy` as follows:

```
BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_TABLE (
    table_name       => 'iceberg_parquet_time_dim3',
    credential_name  => 'OCI_CRED',
    file_uri_list    => '',
    format           =>'{"access_protocol":{"protocol_type":"iceberg",
        "protocol_config":{"iceberg_catalog_type": "hadoop",
        "iceberg_warehouse":"https://objectstorage.uk-
cardiff-1.oraclecloud.com/n/my-tenancy/b/my-iceberg-bucket/o/iceberg",
        "iceberg_table_path": "db.icebergTablePy"}}}');
END;
/
```

Query an Iceberg table on OCI using the location of the root metadata file

We can query the Iceberg table described in the previous section by directly using the URL for the metadata file, as follows:

```
BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_TABLE (
    table_name       => 'iceberg_parquet_time_dim4',
    credential_name  => 'OCI_CRED',
    file_uri_list    => 'https://objectstorage.uk-
cardiff-1.oraclecloud.com/n/my-tenancy/b/my-iceberg-bucket/o/iceberg/db/
icebergTablePy/metadata/v2.metadata.json',
    format           =>'{"access_protocol":{"protocol_type":"iceberg"}}'
    );
  END;
/
```

In this example, we use the `file_uri_list` parameter to specify the URI for the metadata file using the native OCI URI format. When using the metadata file URI, the external table always queries the latest snapshot stored in the specific file. Subsequent updates that generate new snapshots and new metadata files are not accessible to the query.

For more information on the native OCI URI format, see Cloud Object Storage URI Formats.

Validate External Data

To validate any external table, you can use the procedure `DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE`.

To validate a partitioned external table, see Validate External Partitioned Data. This procedure includes a parameter that lets you specify a specific partition to validate.

To validate a hybrid partitioned table, see Validate Hybrid Partitioned Data. This procedure includes a parameter that lets you specify a specific partition to validate.

Before validating an external table you need to create the external table. To create an external table use the procedure for your table type, either `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`. For example:

```
BEGIN
  DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE (
    table_name => 'CHANNELS_EXT' );
END;
/
```

This procedure scans your source files and validates them using the format options specified when you create the external table.

The validate operation, by default, scans all the rows in your source files and stops when a row is rejected. If you want to validate only a subset of the rows, use the `rowcount` parameter. When the `rowcount` parameter is set the validate operation scans rows and stops either when a row is rejected or when the specified number of rows are validated without errors.

For example, the following validate operation scans 100 rows and stops when a row is rejected or when 100 rows are validated without errors:

```
BEGIN
  DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE (
     table_name => 'CHANNELS_EXT',
     rowcount => 100 );
END;
/
```

If you do not want the validate to stop when a row is rejected and you want to see all rejected rows, set the `stop_on_error` parameter to `FALSE`. In this case `VALIDATE_EXTERNAL_TABLE` scans all rows and reports all rejected rows.

If you want to validate only a subset of rows use the `rowcount` parameter. When `rowcount` is set and `stop_on_error` is set to `FALSE`, the validate operation scans rows and stops either when the specified number of rows are rejected or when the specified number of rows are validated without errors. For example, the following example scans 100 rows and stops when 100 rows are rejected or when 100 rows are validated without errors:

```
BEGIN
  DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE (
     table_name => 'CHANNELS_EXT',
     rowcount => 100
     stop_on_error => FALSE );
```

```
END;
/
```

See VALIDATE_EXTERNAL_TABLE Procedure for detailed information about DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE parameters.

See View Logs for Data Validation to see the results of validate operations in the tables dba_load_operations and user_load_operations.

Validate External Partitioned Data

To validate an external partitioned table, you can use the procedure DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE. This procedure includes a parameter that lets you specify a specific partition to validate.

Before validating an external partitioned table you need to create the external partitioned table. To create an external partitioned table use the procedure DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE (see Query External Partitioned Data (with Partitioning Clause) for more details):

```
BEGIN
  DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE (
    table_name => 'PET1',
    partition_name => 'P1');
END;
/
```

This procedure scans your source files for partition P1 and validates them using the format options specified when you create the external partitioned table.

The validation of a partitioned table by default validates all the partitions sequentially until rowcount is reached. If you specify a partition_name then only a specific partition is validated.

The validate operation, by default, scans all the rows in your source files and stops when a row is rejected. If you want to validate only a subset of the rows, use the rowcount parameter. When the rowcount parameter is set the validate operation scans rows and stops either when a row is rejected or when the specified rowcount number of rows are validated without errors.

For example, the following validate operation scans 100 rows and stops when a row is rejected or when 100 rows are validated without errors:

```
BEGIN
  DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE (
     table_name => 'PET1',
     rowcount => 100 );
END;
/
```

If you do not want the validate to stop when a row is rejected and you want to see all rejected rows, set the stop_on_error parameter to FALSE. In this case DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE scans all rows and reports all rejected rows.

If you want to validate only a subset of rows use the rowcount parameter. When rowcount is set and stop_on_error is set to FALSE, the validate operation scans rows and stops either when the specified number of rows are rejected or when the specified number of rows are

validated without errors. For example, the following example scans 100 rows and stops when 100 rows are rejected or when 100 rows are validated without errors:

```
BEGIN
  DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE (
     table_name => 'PET1',
     rowcount => 100
     stop_on_error => FALSE );
END;
/
```

See VALIDATE_EXTERNAL_PART_TABLE Procedure for more information on `DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE`.

See View Logs for Data Validation to see the results of validate operations in the tables `dba_load_operations` and `user_load_operations`.

Validate Hybrid Partitioned Data

To validate a hybrid partitioned table, you can use the procedure `DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE`. This procedure includes a parameter that lets you specify a specific partition to validate.

Before validating a hybrid partitioned table you need to create the table. To create a hybrid partitioned table use the procedure `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` (see Query Hybrid Partitioned Data for more details):

```
BEGIN
 DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE(
    table_name => 'HPT1',
    partition_name => 'P1');
END;
/
```

This procedure scans your source files for partition P1 and validates them using the format options specified when you create the hybrid partitioned table.

The validation of a hybrid partitioned table by default validates all the external partitions sequentially until `rowcount` is reached. If you specify a `partition_name` then only that specific partition is validated.

The validate operation, by default, scans all the rows in your source files and stops when a row is rejected. If you want to validate only a subset of the rows, use the `rowcount` parameter. When the `rowcount` parameter is set the validate operation scans rows and stops either when a row is rejected or when the specified `rowcount` number of rows are validated without errors.

For example, the following validate operation scans 100 rows and stops when a row is rejected or when 100 rows are validated without errors:

```
BEGIN
  DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE (
     table_name => 'HPT1',
     rowcount=>100 );
END;
/
```

If you do not want the validate to stop when a row is rejected and you want to see all rejected rows, set the `stop_on_error` parameter to `FALSE`. In this case `DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE` scans all rows and reports all rejected rows.

If you want to validate only a subset of rows use the `rowcount` parameter. When `rowcount` is set and `stop_on_error` is set to `FALSE`, the validate operation scans rows and stops either when the specified number of rows are rejected or when the specified number of rows are validated without errors. For example, the following example scans 100 rows and stops when 100 rows are rejected or when 100 rows are validated without errors:

```
BEGIN
  DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE (
     table_name => 'HPT1',
     rowcount => 100
     stop_on_error => FALSE );
END;
/
```

For detailed information about `DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE` parameters see VALIDATE_HYBRID_PART_TABLE Procedure.

See View Logs for Data Validation to see the results of validate operations in the tables `dba_load_operations` and `user_load_operations`.

View Logs for Data Validation

To validate an external table, use the procedures `DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE`, `DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE`, and `DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE`.

After you validate your source files you can see the result of the validate operation by querying a load operations table:

- `dba_load_operations`: shows all validate operations.
- `user_load_operations`: shows the validate operations in your schema.

You can use these files to view load validation information. For example use this select operation to query `user_load_operations`:

```
SELECT table_name, owner_name, type, status, start_time, update_time, logfile_table,
badfile_table
FROM user_load_operations
WHERE type = 'VALIDATE';

TABLE_NAME    OWNER_NAME  TYPE       STATUS     START_TIME     UPDATE_TIME
LOGFILE_TABLE    BADFILE_TABLE
------------- ----------  ---------- ---------- -------------- -------------
---------------  ---------------
CHANNELS_EXT  SH          VALIDATE   COMPLETED  13-NOV-17...   13-NOV-17...
VALIDATE$21_LOG  VALIDATE$21_BAD
```

Using this SQL statement with the `WHERE` clause on the `TYPE` column displays all of the load operations with type `VALIDATE`.

The LOGFILE_TABLE column shows the name of the table you can query to look at the log of a validate operation. For example, the following query shows the log for this validate operation:

```
SELECT * FROM VALIDATE$21_LOG;
```

The column BADFILE_TABLE shows the name of the table you can query to look at the rows where there were errors during validation. For example, the following query shows the rejected records for the above validate operation:

```
SELECT * FROM VALIDATE$21_BAD;
```

Depending on the errors shown in the log and the rows shown in the BADFILE_TABLE, you can correct the error by dropping the external table using the DROP TABLE command and recreating it by specifying the correct format options in DBMS_CLOUD.CREATE_EXTERNAL_TABLE, DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE or DBMS_CLOUD.CREATE_HYBRID_PART_TABLE.

> **Note:**
>
> The LOGFILE_TABLE and BADFILE_TABLE tables are stored for two days for each validate operation and then removed automatically.

Use Database Links with Autonomous Database

You can create database links from an Autonomous Database to another Autonomous Database, to other Oracle databases, or to non-Oracle databases. In addition, you can create database links from other databases to an Autonomous Database.

- Create Database Links from Autonomous Database to Another Autonomous Database
  You can create database links from Autonomous Database to another Autonomous Database that is on a private endpoint or on a public endpoint (publicly accessible).
- Create Database Links to an Oracle Database that is not an Autonomous Database
  You can create database links from an Autonomous Database to an Oracle database that is on a private endpoint or on a public endpoint (publicly accessible).
- Create Database Links to Non-Oracle Databases
  You can create database links from Autonomous Database to Non-Oracle databases.
- Create Database Links from Other Databases to Autonomous Database
  You can create database links to an Autonomous Database from an Oracle database that is not an Autonomous Database. For example from an on-premise Oracle database to an Autonomous Database
- Drop Database Links
  After you create a database link you can drop the database link.

Create Database Links from Autonomous Database to Another Autonomous Database

You can create database links from Autonomous Database to another Autonomous Database that is on a private endpoint or on a public endpoint (publicly accessible).

> **Note:**
>
> See Create Database Links to an Oracle Database that is not an Autonomous Database if the target for your database link is an Oracle Database that is not an Autonomous Database.

- Create Database Links from Autonomous Database to a Publicly Accessible Autonomous Database
  You can create database links from an Autonomous Database to a target Autonomous Database that is on a public endpoint.
- Create Database Links from Autonomous Database to an Autonomous Database on a Private Endpoint
  You can create database links from an Autonomous Database to a target Autonomous Database that is on a private endpoint.
- Database Link Notes with a Target that is an Autonomous Database
  Provides notes for creating database links to a target that is another Autonomous Database.

Create Database Links from Autonomous Database to a Publicly Accessible Autonomous Database

You can create database links from an Autonomous Database to a target Autonomous Database that is on a public endpoint.

See How to Create a Database Link from Your Autonomous Database to a Database Cloud Service Instance for more information.

- Create Database Links from Autonomous Database to a Publicly Accessible Autonomous Database without a Wallet (TLS)
  You can create database links from an Autonomous Database to a publicly accessible Autonomous Database without a wallet (TLS).
- Create Database Links from Autonomous Database to a Publicly Accessible Autonomous Database with a Wallet (mTLS)
  You can to create database links from an Autonomous Database to a publicly accessible Autonomous Database with a wallet (mTLS).

Create Database Links from Autonomous Database to a Publicly Accessible Autonomous Database without a Wallet (TLS)

You can create database links from an Autonomous Database to a publicly accessible Autonomous Database without a wallet (TLS).

To create database links to a public target, the target database must be accessible. Some databases, including Autonomous Databases, may limit access (for example, using Access Control Lists). Make sure you enable your target database to allow access from your source database for the database link to work. If you limit access with Access Control Lists (ACLs), you can find the outbound IP address of your source Autonomous Database and allow that IP address to connect to your target database. When the target database is another Autonomous Database, you can add the outbound IP address of the source database to the ACL of the target database.

See Obtain Tenancy Details for information on finding the outbound IP address.

To create a database link to a target Autonomous Database without a wallet (TLS):

1. If you have not already done so, enable TLS connections on your Autonomous Database instance.

   See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for details.

2. On the Autonomous Database instance where you are creating the database link, create credentials to access the target Autonomous Database. The `username` and `password` you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database (you use these credentials to create the database link).

   > **Note:**
   >
   > Supplying the `credential_name` parameter is required.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
           credential_name => 'DB_LINK_CRED',
           username => 'NICK',
           password => 'password' );
   END;
   /
   ```

   The characters in the `username` parameter must be all uppercase letters.

   > **Note:**
   >
   > You can use a vault secret credential for the target database credential in a database link. See Use Vault Secret Credentials for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

3. Create a database link to the target Autonomous Database instance using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

   For example:

   ```
   BEGIN
       DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
           db_link_name => 'SALESLINK',
           hostname => 'adb.eu-frankfurt-1.oraclecloud.com',
           port => '1521',
           service_name => 'example_medium.adb.example.oraclecloud.com',
           credential_name => 'DB_LINK_CRED',
           directory_name => NULL);
   END;
   /
   ```

To create a database link with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to a target Autonomous Database on a public endpoint using a secure TCP connection without a wallet, the `directory_name` parameter must be `NULL`.

The `ssl_server_cert_dn` parameter can either be omitted or if included, specify a `NULL` value.

Users other than ADMIN require privileges to run `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

4. Use the database link you created to access data on the target database.

   For example:

   ```
   SELECT * FROM employees@SALESLINK;
   ```

For the credentials you create in Step 1, the target database credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:

```
BEGIN
    DBMS_CLOUD.UPDATE_CREDENTIAL (
          credential_name => 'DB_LINK_CRED',
          attribute => 'PASSWORD',
          value => 'password' );
END;
/
```

Where *password* is the new password.

After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

> **Note:**
>
> You can create links to Big Data Service using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. See Query Big Data Service Hadoop (HDFS) Data from Autonomous Database for more information.

For additional information, see:

- CREATE_DATABASE_LINK Procedure
- UPDATE_CREDENTIAL Procedure

Create Database Links from Autonomous Database to a Publicly Accessible Autonomous Database with a Wallet (mTLS)

You can to create database links from an Autonomous Database to a publicly accessible Autonomous Database with a wallet (mTLS).

To create database links to a public target, the target database must be accessible. Some databases, including Autonomous Databases, may limit access (for example, using Access Control Lists). Make sure you enable your target database to allow access from your source database for the database link to work. If you limit access with Access Control Lists (ACLs),

you can find the outbound IP address of your source Autonomous Database and allow that IP address to connect to your target database. For example, if the target database is another Autonomous Database, you can add the outbound IP address of the source database to the ACL of the target database.

See Obtain Tenancy Details for information on finding the outbound IP address.

To create database links to a target Autonomous Database with a wallet (mTLS):

1. Copy your target database wallet, `cwallet.sso`, containing the certificates for the target database to Object Store.

   Note the following for the wallet file:

   - The wallet file, along with the Database user ID and password provide access to data in the target Oracle Database. Store wallet files in a secure location. Share wallet files only with authorized users.
   - Do not rename the wallet file. The wallet file in Object Storage must be named `cwallet.sso`.

2. Create credentials to access your Object Store where you store the wallet file `cwallet.sso`. See CREATE_CREDENTIAL Procedure for information about the username and password parameters for different object storage services.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Create a directory on Autonomous Database for the wallet file `cwallet.sso`.

   For example:

   ```
   CREATE DIRECTORY dblink_wallet_dir AS 'directory_path_of_your_choice';
   ```

   See Create Directory in Autonomous Database for information on creating directories.

4. Use `DBMS_CLOUD.GET_OBJECT` to upload the target database wallet to the directory you created in the previous step, *DBLINK_WALLET_DIR*.

   For example:

   ```
   BEGIN
       DBMS_CLOUD.GET_OBJECT(
           credential_name => 'DEF_CRED_NAME',
           object_uri => 'https://objectstorage.us-
   phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/cwallet.sso',
           directory_name => 'DBLINK_WALLET_DIR');
   END;
   /
   ```

   In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.

> **Note:**
>
> The `credential_name` you use in this step is the credentials for the Object Store. In the next step you create the credentials to access the target database.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

5. On the Autonomous Database instance, create credentials to access the target database. The `username` and `password` you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database that you use to create the database link.

> **Note:**
>
> Supplying the `credential_name` parameter is required.

For example:

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'DB_LINK_CRED',
        username => 'NICK',
        password => 'password');
END;
/
```

The characters in the `username` parameter must be all uppercase letters.

> **Note:**
>
> You can use a vault secret credential for the target database credential in a database link. See Use Vault Secret Credentials for more information.

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

6. Create the database link to the target database using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

For example:

```
BEGIN
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'SALESLINK',
        hostname => 'adb.eu-frankfurt-1.oraclecloud.com',
        port => '1522',
        service_name => 'example_medium.adb.example.oraclecloud.com',
        ssl_server_cert_dn => 'CN=adb.example.oraclecloud.com,OU=Oracle
BMCS FRANKFURT,O=Oracle Corporation,L=Redwood City,ST=California,C=US',
        credential_name => 'DB_LINK_CRED',
```

```
        directory_name => 'DBLINK_WALLET_DIR');
END;
/
```

Users other than ADMIN require privileges to run `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

If the wallet file in the directory specified with `directory_name` is not `cwallet.sso`, the procedure reports an error such as: `ORA-28759: failure to open file`.

7. Use the database link you created to access data on the target database.

   For example:

   ```
   SELECT * FROM employees@SALESLINK;
   ```

For the credentials you create in Step 5, the target database credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:

```
BEGIN
    DBMS_CLOUD.UPDATE_CREDENTIAL (
        credential_name => 'DB_LINK_CRED',
        attribute => 'PASSWORD',
        value => 'password' );
END;
/
```

Where *password* is the new password.

After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

> **Note:**
>
> You can create links to Big Data Service using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. See Query Big Data Service Hadoop (HDFS) Data from Autonomous Database for more information.

For additional information, see:

- CREATE_DATABASE_LINK Procedure
- GET_OBJECT Procedure and Function
- UPDATE_CREDENTIAL Procedure

Create Database Links from Autonomous Database to an Autonomous Database on a Private Endpoint

You can create database links from an Autonomous Database to a target Autonomous Database that is on a private endpoint.

> **Note:**
>
> Database links from an Autonomous Database to a target Autonomous Database that is on a private endpoint are only supported in commercial regions and US Government regions.
> This feature is enabled by default in all commercial regions.
>
> This feature is enabled by default in US Government regions for newly provisioned databases.
>
> For existing US Government databases on a private endpoint, if you want to create database links from an Autonomous Database to a target in a US Government region, you can file a Service Request at Oracle Cloud Support and request to enable the private endpoint in government regions database linking feature.
>
> US Government regions include the following:
>
> - Oracle Cloud Infrastructure US Government Cloud with FedRAMP Authorization
> - Oracle Cloud Infrastructure US Federal Cloud with DISA Impact Level 5 Authorization

Depending on configuration of the target Autonomous Database, you have these options:

- Create Database Links to a Target Autonomous Database on a Private Endpoint without a Wallet (TLS)
- Create Database Links to a Target Autonomous Database on a Private Endpoint with a Wallet (mTLS)

See How to Create a Database Link from Your Autonomous Database to a Database Cloud Service Instance for more information.

Topics

- Prerequisites for Database Links from Autonomous Database to a Target Autonomous Database on a Private Endpoint
  Lists the prerequisites to create database links to a target Autonomous Database that is on a private endpoint.
- Create Database Links to a Target Autonomous Database on a Private Endpoint without a Wallet (TLS)
  You can create database links from an Autonomous Database to a target Autonomous Database that is on a private endpoint and connect without a wallet (TLS).
- Create Database Links to a Target Autonomous Database on a Private Endpoint with a Wallet (mTLS)
  You can create database links from an Autonomous Database to a target Autonomous Database that is on a private endpoint (mTLS).

Prerequisites for Database Links from Autonomous Database to a Target Autonomous Database on a Private Endpoint

Lists the prerequisites to create database links to a target Autonomous Database that is on a private endpoint.

To create a database link to a target Autonomous Database on a private endpoint:

- The target database must be accessible from the source database's Oracle Cloud Infrastructure VCN. For example, you can connect to the target database when:
  - The target database is on a private endpoint.
  - Both the source database and the target database are in the same Oracle Cloud Infrastructure VCN.
  - The source database and the target database are in different Oracle Cloud Infrastructure VCNs that are paired.
  - The target database is connected to the source database's Oracle Cloud Infrastructure VCN using FastConnect or VPN.
- For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a SCAN IP, or a SCAN hostname).
- `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` does not support a value of `localhost` for the `hostname` parameter.
- The following ingress and egress rules must be defined for the private endpoint:
  - Define an egress rule in the source database's subnet security list or network security group such that the traffic over TCP is allowed to the target database's IP address and port number.
  - Define an ingress rule in the target database's subnet security list or network security group such that the traffic over TCP is allowed from the source database IP address to the destination port.

  See Configure Network Access with Private Endpoints for information on configuring private endpoints with ingress and egress rules.

> **Note:**
>
> When your Autonomous Database instance is configured with a private endpoint, set the `ROUTE_OUTBOUND_CONNECTIONS` database property to '`PRIVATE_ENDPOINT`' to specify that all outgoing database links are subject to the Autonomous Database instance private endpoint VCN's egress rules. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

Create Database Links to a Target Autonomous Database on a Private Endpoint without a Wallet (TLS)

You can create database links from an Autonomous Database to a target Autonomous Database that is on a private endpoint and connect without a wallet (TLS).

Perform the prerequisite steps, as required. See Prerequisites for Database Links from Autonomous Database to a Target Autonomous Database on a Private Endpoint for details.

To create a database link to a target Autonomous Database on a private endpoint without a wallet:

1. If you have not already done so, enable TLS connections on your Autonomous Database instance.

   See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for details.

2. Create credentials to access the target Autonomous Database instance. The `username` and `password` you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database used within the database link, (where the target database is accessed through the VCN).

   For example:

   ```
   BEGIN
        DBMS_CLOUD.CREATE_CREDENTIAL(
            credential_name => 'PRIVATE_ENDPOINT_CRED',
            username => 'NICK',
            password => 'password'
            );
   END;
   /
   ```

   The characters in the `username` parameter must be all uppercase letters.

   > **Note:**
   >
   > You can use a vault secret credential for the target database credential in a database link. See Use Vault Secret Credentials for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

3. Create the database link to the target database using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

   For example:

   ```
   BEGIN
        DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
            db_link_name => 'PRIVATE_ENDPOINT_LINK',
            hostname => 'exampleHostname',
            port => '1521',
            service_name => 'example_high.adb.oraclecloud.com',
   ```

```
        credential_name => 'PRIVATE_ENDPOINT_CRED',
        directory_name => NULL,
        private_target => TRUE);
END;
/
```

For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a SCAN IP, or a SCAN hostname).

Users other than ADMIN require privileges to run `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

As shown in the example, to create a database link with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to a target database on a private endpoint without a wallet, all of the following are required:

- The `directory_name` parameter must be `NULL`.
- The `ssl_server_cert_dn` parameter can either be omitted or if included, specify a `NULL` value.
- The `private_target` parameter must be `TRUE`.

> **Note:**
>
> If you set `ROUTE_OUTBOUND_CONNECTIONS` to `PRIVATE_ENDPOINT`, setting the `private_target` parameter to `TRUE` is not required in this API. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

4. Use the database link you created to access data in the target database.

   For example:

```
SELECT * FROM employees@PRIVATE_ENDPOINT_LINK;
```

> **Note:**
>
> For the credentials you create in Step 1, the Oracle Database credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:
>
> ```
> BEGIN
>      DBMS_CLOUD.UPDATE_CREDENTIAL (
>          credential_name => 'DB_LINK_CRED',
>          attribute => 'PASSWORD',
>          value => 'password');
> END;
> /
> ```
>
> Where *password* is the new password.
>
> After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

See CREATE_DATABASE_LINK Procedure for additional information.

Create Database Links to a Target Autonomous Database on a Private Endpoint with a Wallet (mTLS)

You can create database links from an Autonomous Database to a target Autonomous Database that is on a private endpoint (mTLS).

Perform the prerequisite steps, as required. See Prerequisites for Database Links from Autonomous Database to a Target Autonomous Database on a Private Endpoint for details.

To create a database link to a target Autonomous Database on a private endpoint, with a wallet:

1. Copy your target database wallet, `cwallet.sso`, containing the certificates for the target database to Object Store.

   > **Note:**
   >
   > The wallet file, along with the Database user ID and password provide access to data in the target Oracle database. Store wallet files in a secure location. Share wallet files only with authorized users.

2. Create credentials to access your Object Store where you store the `cwallet.sso`. See CREATE_CREDENTIAL Procedure for information about the username and password parameters for different object storage services.
3. Create a directory on Autonomous Database for the wallet file `cwallet.sso`.

   For example:

   ```
   CREATE DIRECTORY wallet_dir AS 'directory_path_of_your_choice';
   ```

See Create Directory in Autonomous Database for information on creating directories.

4. Use `DBMS_CLOUD.GET_OBJECT` to upload the target database wallet to the directory you created in the previous step, *WALLET_DIR*.

   For example:

```
BEGIN
    DBMS_CLOUD.GET_OBJECT(
        credential_name => 'DEF_CRED_NAME',
        object_uri => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/cwallet.sso',
        directory_name => 'WALLET_DIR');
END;
/
```

   In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.

> **Note:**
>
> The `credential_name` you use in this step is the credentials for the Object Store. In the next step you create the credentials to access the target database.

5. On Autonomous Database create credentials to access the target database. The `username` and `password` you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database used within the database link, (where the target database is accessed through the VCN).

> **Note:**
>
> Supplying the `credential_name` parameter is required.

   For example:

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DB_LINK_CRED',
       username => 'NICK',
       password => 'password');
END;
/
```

   The characters in the `username` parameter must be all uppercase letters.

> **Note:**
>
> You can use a vault secret credential for the target database credential in a database link. See Use Vault Secret Credentials for more information.

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

6. Create the database link to the target database using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

   For example:

```
BEGIN
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'PEDBLINK1',
        hostname => 'example1.adb.ap-osaka-1.oraclecloud.com',
        port => '1522',
        service_name => 'example_high.adb.oraclecloud.com',
        ssl_server_cert_dn => 'ssl_server_cert_dn',
        credential_name => 'DB_LINK_CRED',
        directory_name => 'WALLET_DIR',
        private_target => TRUE);
END;
/
```

> **Note:**
>
> If you set `ROUTE_OUTBOUND_CONNECTIONS` to `PRIVATE_ENDPOINT`, setting the `private_target` parameter to `TRUE` is not required in this API. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

   For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a SCAN IP, or a SCAN hostname).

   `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` does not support a value of `localhost` for the `hostname` parameter.

   Users other than ADMIN require privileges to run `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

7. Use the database link you created to access data in the target database.

   For example:

```
SELECT * FROM employees@PEDBLINK1;
```

> **Note:**
>
> For the credentials you create in Step 5, the Oracle Database credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:
>
> ```
> BEGIN
>     DBMS_CLOUD.UPDATE_CREDENTIAL (
>         credential_name => 'DB_LINK_CRED',
>         attribute => 'PASSWORD',
>         value => 'password');
> END;
> /
> ```
>
> Where *password* is the new password.
>
> After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

See CREATE_DATABASE_LINK Procedure for additional information.

Database Link Notes with a Target that is an Autonomous Database

Provides notes for creating database links to a target that is another Autonomous Database.

Notes for database links to another Autonomous Database:

- Only one wallet file is valid per directory for use with database links. You can only upload one `cwallet.sso` at a time to the directory you choose for wallet files (for example *DBLINK_WALLET_DIR*). This means with a `cwallet.sso` in *DBLINK_WALLET_DIR* you can only create database links to the databases for which the wallet in that directory is valid. To use multiple `cwallet.sso` files with database links you need to create additional directories and put each `cwallet.sso` in a different directory. When you create database links with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`, specify the directory that contains the wallet with the `directory_name` parameter.

  See Create Directory in Autonomous Database for information on creating directories.
- To list the database links, use the `ALL_DB_LINKS` view. See ALL_DB_LINKS for more information.
- The wallet file, along with the Database user ID and password provide access to data in the target Autonomous Database. Store wallet files in a secure location. Share wallet files only with authorized users.

Create Database Links to an Oracle Database that is not an Autonomous Database

You can create database links from an Autonomous Database to an Oracle database that is on a private endpoint or on a public endpoint (publicly accessible).

> **Note:**
>
> See Create Database Links from Autonomous Database to Another Autonomous Database if the target for your database link is another Autonomous Database.

- Create Database Links from Autonomous Database to a Publicly Accessible Oracle Database with a Wallet (mTLS)
  You can create database links from an Autonomous Database to a target Oracle database that is on a public endpoint.
- Create Database Links from Autonomous Database to an Oracle Database on a Private Endpoint
  You can create database links from an Autonomous Database to a target Oracle Database that is on a private endpoint.
- Database Link Notes with a Target Oracle Database
  Provides notes for creating database links to a target Oracle database (when the target is not an Autonomous Database)

Create Database Links from Autonomous Database to a Publicly Accessible Oracle Database with a Wallet (mTLS)

You can create database links from an Autonomous Database to a target Oracle database that is on a public endpoint.

To use database links with Autonomous Database the target database must be configured to use TCP/IP with SSL (TCPS) authentication. Autonomous Databases use TCP/IP with SSL (TCPS) authentication by default, so you do not need to do any additional configuration in your target database to link to another Autonomous Database. Other Oracle databases must be configured to use TCP/IP with SSL (TCPS) authentication. See Configuring Secure Sockets Layer Authentication for more information.

To create database links to a public target, the target Oracle Database must be accessible. Some databases may limit access (for example, using Access Control Lists). Make sure you enable your target database to allow access from your source database for the database link to work. If you limit access with Access Control Lists (ACLs), you can find the outbound IP address of your source Autonomous Database and allow that IP address to connect to your target database.

See How to Create a Database Link from Your Autonomous Database to a Database Cloud Service Instance for more information.

To create database links to a target Oracle database with a wallet (mTLS):

1. Copy your target database wallet, `cwallet.sso`, containing the certificates for the target database to Object Store.

   Note the following for the wallet file:

   - The wallet file, along with the Database user ID and password provide access to data in the target Oracle Database. Store wallet files in a secure location. Share wallet files only with authorized users.
   - Do not rename the wallet file. The wallet file in Object Storage must be named `cwallet.sso`.

2. Create credentials to access your Object Store where you store the wallet file `cwallet.sso`. See CREATE_CREDENTIAL Procedure for information about the username and password parameters for different object storage services.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Create a directory on Autonomous Database for the wallet file `cwallet.sso`.

   For example:

   ```
   CREATE DIRECTORY dblink_wallet_dir AS 'directory_path_of_your_choice';
   ```

   See Create Directory in Autonomous Database for information on creating directories.

4. Use `DBMS_CLOUD.GET_OBJECT` to upload the target database wallet to the directory you created in the previous step, *DBLINK_WALLET_DIR*.

   For example:

   ```
   BEGIN
       DBMS_CLOUD.GET_OBJECT(
           credential_name => 'DEF_CRED_NAME',
           object_uri => 'https://objectstorage.us-
   phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/cwallet.sso',
           directory_name => 'DBLINK_WALLET_DIR');
   END;
   /
   ```

   In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.

   > **Note:**
   >
   > The `credential_name` you use in this step is the credentials for the Object Store. In the next step you create the credentials to access the target database.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

5. On the Autonomous Database instance, create credentials to access the target database. The `username` and `password` you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database that you use to create the database link.

   > **Note:**
   >
   > Supplying the `credential_name` parameter is required.

For example:

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'DB_LINK_CRED',
        username => 'NICK',
        password => 'password');
END;
/
```

The characters in the `username` parameter must be all uppercase letters.

> **Note:**
>
> You can use a vault secret credential for the target database credential in a database link. See Use Vault Secret Credentials for more information.

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

6. Create the database link to the target database using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

   For example:

```
BEGIN
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'SALESLINK',
        hostname => 'adb.eu-frankfurt-1.oraclecloud.com',
        port => '1522',
        service_name => 'example_medium.adb.example.oraclecloud.com',
        ssl_server_cert_dn => 'CN=adb.example.oraclecloud.com,OU=Oracle
BMCS FRANKFURT,O=Oracle Corporation,L=Redwood City,ST=California,C=US',
        credential_name => 'DB_LINK_CRED',
        directory_name => 'DBLINK_WALLET_DIR');
END;
/
```

   Users other than ADMIN require privileges to run `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

   If the wallet file in the directory specified with `directory_name` is not `cwallet.sso`, the procedure reports an error such as: `ORA-28759: failure to open file`.

7. Use the database link you created to access data on the target database.

   For example:

```
SELECT * FROM employees@SALESLINK;
```

For the credentials you create in Step 5, the target database credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:

```
BEGIN
    DBMS_CLOUD.UPDATE_CREDENTIAL (
        credential_name => 'DB_LINK_CRED',
        attribute => 'PASSWORD',
        value => 'password' );
END;
/
```

Where *password* is the new password.

After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

> **Note:**
>
> You can create links to Big Data Service using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. See Query Big Data Service Hadoop (HDFS) Data from Autonomous Database for more information.

For additional information, see:

- CREATE_DATABASE_LINK Procedure
- GET_OBJECT Procedure and Function
- UPDATE_CREDENTIAL Procedure

Create Database Links from Autonomous Database to an Oracle Database on a Private Endpoint

You can create database links from an Autonomous Database to a target Oracle Database that is on a private endpoint.

> **Note:**
>
> Database links from an Autonomous Database to a target Oracle database that is on a private endpoint are only supported in commercial regions and US Government regions.
> This feature is enabled by default in all commercial regions.
>
> This feature is enabled by default in US Government regions for newly provisioned databases.
>
> For existing US Government databases on a private endpoint, if you want to create database links from an Autonomous Database to a target in a US Government region, you can file a Service Request at Oracle Cloud Support and request to enable the private endpoint in government regions database linking feature.
>
> US Government regions include the following:
>
> - Oracle Cloud Infrastructure US Government Cloud with FedRAMP Authorization
> - Oracle Cloud Infrastructure US Federal Cloud with DISA Impact Level 5 Authorization

Depending on the type and the configuration of the target Oracle database:

- **Another Oracle Database, such as on-premises or a Database Cloud Service database, on a private endpoint that is configured for SSL (TCPS)**: In this case you can create the database link with a wallet, and the database link communicates with TCPS. See Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint with a Wallet (mTLS) for details:
- **Oracle Database, such as on-premises or a Database Cloud Service database, on a private endpoint that is configured for TCP**: In this case you create the database link without a wallet and the database link communicates with TCP. See Create Database Links to Oracle Databases on a Private Endpoint without a Wallet for details

See How to Create a Database Link from Your Autonomous Database to a Database Cloud Service Instance for more information.

Topics

- Prerequisites for Database Links from Autonomous Database to Oracle Databases on a Private Endpoint
  Lists the prerequisites to create database links from an Autonomous Database to a target Oracle database that is on a private endpoint.
- Create Database Links to Oracle Databases on a Private Endpoint without a Wallet
  Use `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to create database links from an Autonomous Database to a target Oracle database that is on a private endpoint and connect without a wallet (TCP).

- Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint with a Wallet (mTLS)
  You can create database links from an Autonomous Database to a target Oracle database that is on a private endpoint.

Prerequisites for Database Links from Autonomous Database to Oracle Databases on a Private Endpoint

Lists the prerequisites to create database links from an Autonomous Database to a target Oracle database that is on a private endpoint.

To create a database link to a target Oracle database on a private endpoint:

- The target database must be accessible from the source database's Oracle Cloud Infrastructure VCN. For example, you can connect to the target database when:
  - The target database is on a private endpoint.
  - Both the source database and the target database are in the same Oracle Cloud Infrastructure VCN.
  - The source database and the target database are in different Oracle Cloud Infrastructure VCNs that are paired.
  - The target database is an on-premises database that is connected to the source database's Oracle Cloud Infrastructure VCN using FastConnect or VPN.
- There are two options to specify the target database, use the `hostname` parameter or the `rac_hostnames` parameter:
  - For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a SCAN IP, or a SCAN hostname).
  - When the target is an Oracle RAC database, use the `rac_hostnames` parameter to specify one or more hostnames with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. This allows you to take advantage of the high availability capabilities of Oracle RAC. Using an IP address, a SCAN IP, or a SCAN hostname in the `rac_hostnames` value is not supported.

    When you specify a list of host names in the `rac_hostnames` parameter, `CREATE_DATABASE_LINK` uses all of the specified host names as addresses in the connect string. If one of the specified hosts is not available on the target Oracle RAC database, Autonomous Database automatically attempts to connect using another host name from the list.
- The following ingress and egress rules must be defined for the private endpoint:
  - Define an egress rule in the source database's subnet security list or network security group such that the traffic over TCP is allowed to the target database's IP address and port number.
  - Define an ingress rule in the target database's subnet security list or network security group such that the traffic over TCP is allowed from the source database IP address to the destination port.

  See Configure Network Access with Private Endpoints for information on configuring private endpoints with ingress and egress rules.

> **Note:**
>
> When your Autonomous Database instance is configured with a private endpoint, set the `ROUTE_OUTBOUND_CONNECTIONS` database parameter to '`PRIVATE_ENDPOINT`' to specify that all outgoing database links are subject to the Autonomous Database instance private endpoint VCN's egress rules. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

Create Database Links to Oracle Databases on a Private Endpoint without a Wallet

Use `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to create database links from an Autonomous Database to a target Oracle database that is on a private endpoint and connect without a wallet (TCP).

> **Note:**
>
> This option is for target Oracle databases that are on a private endpoint and do not have SSL/TCPS configured.

Perform the prerequisite steps, as required. See Prerequisites for Database Links from Autonomous Database to a Target Autonomous Database on a Private Endpoint for details.

To create a database link to a target database on a private endpoint using a secure TCP connection without a wallet:

1. On Autonomous Database create credentials to access the target database. The `username` and `password` you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database used within the database link, (where the target database is accessed through the VCN).

   For example:

   ```
   BEGIN
        DBMS_CLOUD.CREATE_CREDENTIAL(
            credential_name => 'PRIVATE_ENDPOINT_CRED',
            username => 'NICK',
            password => 'password'
            );
   END;
   /
   ```

   The characters in the `username` parameter must be all uppercase letters.

   > **Note:**
   >
   > You can use a vault secret credential for the target database credential in a database link. See Use Vault Secret Credentials for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

2. Create the database link to the target database using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

   For example:

```
BEGIN
     DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
         db_link_name => 'PRIVATE_ENDPOINT_LINK',
         hostname => 'exampleHostname',
         port => '1522',
         service_name => 'exampleServiceName',
         ssl_server_cert_dn => NULL,
         credential_name => 'PRIVATE_ENDPOINT_CRED',
         directory_name => NULL,
         private_target => TRUE);
END;
/
```

   For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a SCAN IP, or a SCAN hostname).

   When the target is an Oracle RAC database, use the `rac_hostnames` parameter to specify one or more hostnames with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. This allows you to take advantage of the high availability capabilities of Oracle RAC. Using an IP address, a SCAN IP, or a SCAN hostname in the `rac_hostnames` value is not supported.

   For example, with a target Oracle RAC database use the `rac_hostnames` parameter:

```
BEGIN
     DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
         db_link_name => 'PRIVATE_ENDPOINT_LINK',
         rac_hostnames => '["sales1-svr1.example.adb.us-
ashburn-1.oraclecloud.com",
                           "sales1-svr2.example.adb.us-
ashburn-1.oraclecloud.com",
                           "sales1-svr3.example.adb.us-
ashburn-1.oraclecloud.com"]',
         port => '1522',
         service_name => 'exampleServiceName',
         ssl_server_cert_dn => NULL,
         credential_name => 'PRIVATE_ENDPOINT_CRED',
         directory_name => NULL,
         private_target => TRUE);
END;
/
```

   `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` does not support a value of `localhost` for the `hostname` or in the `rac_hostnames` parameter.

   Users other than ADMIN require privileges to run `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

As shown in the example, to create a database link with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to a target database on a private endpoint using a secure TCP connection without a wallet, all of the following are required:

- The `directory_name` parameter must be `NULL`.
- The `ssl_server_cert_dn` parameter must be `NULL`.
- The `private_target` parameter must be `TRUE`.

> **Note:**
>
> If you set `ROUTE_OUTBOUND_CONNECTIONS` to `PRIVATE_ENDPOINT`, setting the `private_target` parameter to `TRUE` is not required in this API. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

3. Use the database link you created to access data in the target database.

   For example:

```
SELECT * FROM employees@PRIVATE_ENDPOINT_LINK;
```

> **Note:**
>
> For the credentials you create in Step 1, the Oracle Database credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:
>
> ```
> BEGIN
>      DBMS_CLOUD.UPDATE_CREDENTIAL (
>          credential_name => 'DB_LINK_CRED',
>          attribute => 'PASSWORD',
>          value => 'password');
> END;
> /
> ```
>
> Where *password* is the new password.
>
> After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

See CREATE_DATABASE_LINK Procedure for additional information.

Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint with a Wallet (mTLS)

You can create database links from an Autonomous Database to a target Oracle database that is on a private endpoint.

> **Note:**
>
> This option is for target Oracle databases that have SSL/TCPS configured and that are on a private endpoint.

If the target Oracle database does not have SSL/TCPS configured, you have two options:

- You can configure the target Oracle database to use TCP/IP with SSL (TCPS) authentication. See Configuring Transport Layer Security Authentication for information on configuring SSL/TCPS.
- You can connect to the target Oracle database with TCP. See Create Database Links to Oracle Databases on a Private Endpoint without a Wallet for details.

Perform the prerequisite steps, as required. See Prerequisites for Database Links from Autonomous Database to a Target Autonomous Database on a Private Endpoint for details.

To create a database link to a target Oracle database on a private endpoint using TCP/IP with SSL (TCPS) authentication:

1. Copy your target database wallet, `cwallet.sso`, containing the certificates for the target database to Object Store.

   > **Note:**
   >
   > The wallet file, along with the Database user ID and password provide access to data in the target Oracle database. Store wallet files in a secure location. Share wallet files only with authorized users.

2. Create credentials to access your Object Store where you store the `cwallet.sso`. See CREATE_CREDENTIAL Procedure for information about the username and password parameters for different object storage services.
3. Create a directory on Autonomous Database for the wallet file `cwallet.sso`.

   For example:

   ```
   CREATE DIRECTORY wallet_dir AS 'directory_path_of_your_choice';
   ```

   See Create Directory in Autonomous Database for information on creating directories.
4. Use `DBMS_CLOUD.GET_OBJECT` to upload the target database wallet to the directory you created in the previous step, *WALLET_DIR*.

   For example:

   ```
   BEGIN
       DBMS_CLOUD.GET_OBJECT(
   ```

```
        credential_name => 'DEF_CRED_NAME',
        object_uri => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/cwallet.sso',
        directory_name => 'WALLET_DIR');
END;
/
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.

> **Note:**
>
> The `credential_name` you use in this step is the credentials for the Object Store. In the next step you create the credentials to access the target database.

5. On Autonomous Database create credentials to access the target database. The `username` and `password` you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database used within the database link, (where the target database is accessed through the VCN).

> **Note:**
>
> Supplying the `credential_name` parameter is required.

For example:

```
BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
      credential_name => 'DB_LINK_CRED',
      username => 'NICK',
      password => 'password');
END;
/
```

The characters in the `username` parameter must be all uppercase letters.

> **Note:**
>
> You can use a vault secret credential for the target database credential in a database link. See Use Vault Secret Credentials for more information.

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

6. Create the database link to the target database using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

For example:

```
BEGIN
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'PEDBLINK1',
        hostname => 'example1.adb.ap-osaka-1.oraclecloud.com',
        port => '1522',
        service_name => 'example_high.adb.oraclecloud.com',
        ssl_server_cert_dn => 'ssl_server_cert_dn',
        credential_name => 'DB_LINK_CRED',
        directory_name => 'WALLET_DIR',
        private_target => TRUE);
END;
/
```

> **Note:**
>
> If you set `ROUTE_OUTBOUND_CONNECTIONS` to `PRIVATE_ENDPOINT`, setting the `private_target` parameter to `TRUE` is not required in this API. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a SCAN IP, or a SCAN hostname).

When the target is an Oracle RAC database, use the `rac_hostnames` parameter to specify one or more hostnames with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. This allows you to take advantage of the high availability capabilities of Oracle RAC. Using an IP address, a SCAN IP, or a SCAN hostname in the `rac_hostnames` value is not supported.

For example, with a target Oracle RAC database use the `rac_hostnames` parameter:

```
BEGIN
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'PEDBLINK1',
        rac_hostnames => '["sales1-svr1.example.adb.us-
ashburn-1.oraclecloud.com",
                          "sales1-svr2.example.adb.us-
ashburn-1.oraclecloud.com",
                          "sales1-svr3.example.adb.us-
ashburn-1.oraclecloud.com"]',
        port => '1522',
        service_name => 'example_high.adb.oraclecloud.com',
        ssl_server_cert_dn => 'ssl_server_cert_dn',
        credential_name => 'DB_LINK_CRED',
        directory_name => 'WALLET_DIR',
        private_target => TRUE);
END;
/
```

`DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` does not support a value of `localhost` for the `hostname` or in the `rac_hostnames` parameter.

Users other than ADMIN require privileges to run `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

7. Use the database link you created to access data in the target database.

   For example:

```
SELECT * FROM employees@PEDBLINK1;
```

> **Note:**
>
> For the credentials you create in Step 5, the Oracle Database credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:
>
> ```
> BEGIN
>     DBMS_CLOUD.UPDATE_CREDENTIAL (
>         credential_name => 'DB_LINK_CRED',
>         attribute => 'PASSWORD',
>         value => 'password');
> END;
> /
> ```
>
> Where *password* is the new password.
>
> After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

See CREATE_DATABASE_LINK Procedure for additional information.

Database Link Notes with a Target Oracle Database

Provides notes for creating database links to a target Oracle database (when the target is not an Autonomous Database)

Notes for database links to other Oracle databases:

- If you are using database links between Autonomous Database and other Oracle Databases, you might need to apply Patch 33843368 on the Oracle Database that is not an Autonomous Database. This applies to cases where the Autonomous Database instance is either the source or the target of the database link.

  See My Oracle Support Knowledge Base: Patch Requirement For Database Links Between ADB-S And Other Oracle Databases(Doc ID 2874244.1) for further details.

- Only one wallet file is valid per directory for use with database links. You can only upload one `cwallet.sso` at a time to the directory you choose for wallet files (for example *DBLINK_WALLET_DIR*). This means with a `cwallet.sso` in *DBLINK_WALLET_DIR* you can only create database links to the databases for which the wallet in that directory is valid. To use multiple `cwallet.sso` files with database links you need to create additional directories and put each `cwallet.sso` in a different directory. When you create database

links with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`, specify the directory that contains the wallet with the `directory_name` parameter.

See Create Directory in Autonomous Database for information on creating directories.

- Supported target Oracle database versions for database links to another Oracle Database are: 19c, 12.2.0, and 12.1.0.

> **Note:**
>
> For complete information on supported versions, see Client Server Interoperability Support Matrix for Different Oracle Versions (Doc ID 207303.1)

- Autonomous Database sets the `SEC_CASE_SENSITIVE_LOGON` parameter to `true` and this value cannot be changed. If your target database is not an Autonomous Database, then you must set `SEC_CASE_SENSITIVE_LOGON` parameter to `true` on the target database. If `SEC_CASE_SENSITIVE_LOGON` is set to `false` on the target database, then error `ORA-28040: No matching authentication protocol` is raised.
- To list the database links, use the `ALL_DB_LINKS` view. See ALL_DB_LINKS for more information.
- The wallet file, along with the Database user ID and password provide access to data in the target Oracle database. Store wallet files in a secure location. Share wallet files only with authorized users.
- When the Autonomous Database instance is on a private endpoint, there are two options to specify the target database: use either the `hostname` parameter or the `rac_hostnames` parameter:
  - For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a SCAN IP, or a SCAN hostname).
  - When the target is an Oracle RAC database, use the `rac_hostnames` parameter to specify one or more hostnames with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. This allows you to take advantage of the high availability capabilities of Oracle RAC. Using an IP address, a SCAN IP, or a SCAN hostname in the `rac_hostnames` value is not supported.

    When you specify a list of host names in the `rac_hostnames` parameter, `CREATE_DATABASE_LINK` uses all of the specified host names as addresses in the connect string. If one of the specified hosts is not available on the target Oracle RAC database, Autonomous Database automatically attempts to connect using another host name from the list.
  - `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` does not support a value of `localhost` for the `hostname` or in the `rac_hostnames` parameter.

Create Database Links to Non-Oracle Databases

You can create database links from Autonomous Database to Non-Oracle databases.

- Create Database Links to Non-Oracle Databases with Oracle-Managed Heterogeneous Connectivity
  Autonomous Database support for Oracle-managed heterogeneous connectivity makes it easy to create database links to non-Oracle databases. When you use database links with Oracle-managed heterogeneous connectivity, Autonomous Database configures and sets up the connection to the non-Oracle database.
- Create Database Links to Non-Oracle Databases with Customer-Managed Heterogeneous Connectivity
  Use `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to create database links from an Autonomous Database to an Oracle Database Gateway to access Non-Oracle databases.

Create Database Links to Non-Oracle Databases with Oracle-Managed Heterogeneous Connectivity

Autonomous Database support for Oracle-managed heterogeneous connectivity makes it easy to create database links to non-Oracle databases. When you use database links with Oracle-managed heterogeneous connectivity, Autonomous Database configures and sets up the connection to the non-Oracle database.

The following is a prerequisite to use Oracle-managed heterogeneous connectivity with Autonomous Database:

- When the target database is on a public endpoint, database must be configured to allow incoming SSL/TLS connections with CA signed certificates.

  Oracle-managed heterogeneous connectivity supports connections to target database services on private endpoints (for example you can connect to Oracle MySQL Database Service when the service is on a private endpoint). When you connect to a non-oracle database with Oracle-managed heterogeneous connectivity on a private endpoint, the connection uses TCP protocol and it does not require SSL/TLS to be configured on the target database.

  See Create Database Links to Oracle MySQL on a Private Endpoint with Oracle-Managed Heterogeneous Connectivity for more information.

To create database links to a non-Oracle database using Oracle-managed heterogeneous connectivity, do the following:

1. On Autonomous Database create credentials to access the target database. The `username` and `password` you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database used within the database link.

   > **Note:**
   >
   > Supplying the `credential_name` parameter is required.

   For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'AWS_REDSHIFT_LINK_CRED',
    username => 'nick',
    password => 'password'
  );
```

```
END;
/
```

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

To access Google Analytics, Google BigQuery, or ServiceNow with OAuth2, the credential must include the `params` parameter with the value `gcp_oauth2`.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'SERVICENOW_OAUTH',
    params => JSON_OBJECT(
                'gcp_oauth2' value JSON_OBJECT(
                       'client_id' value 'CLIENT_ID',
                       'client_secret' value 'CLIENT_SECRET',
                       'refresh_token' value 'Refresh_Token')));
END;
/
```

See CREATE_CREDENTIAL Procedure for more information.

2. Create the database link to the target database using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

For example, to create a database link to AWS Redshift:

```
BEGIN
     DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
          db_link_name => 'AWSREDSHIFT_LINK',
          hostname => 'example.com',
          port => '5439',
          service_name => 'example_service_name',
          credential_name => 'AWS_REDSHIFT_LINK_CRED',
          gateway_params => JSON_OBJECT('db_type'  value 'awsredshift'),
          ssl_server_cert_dn => NULL);
END;
/
```

The `service_name` is the database name of the non-Oracle database.

The `gateway_params db_type` value that you supply must be one of the supported values:

db_type Value	Database Type
`awsredshift`	Amazon Redshift
`azure`	Microsoft SQL Server Azure SQL Azure Synapse Analytics
`db2`	IBM Db2
`google_analytics`	Google Analytics
`google_bigquery`	Google BigQuery

db_type Value	Database Type
hive	Apache Hive
mongodb	MongoDB
mysql	MySQL
mysql_community	MySQL Community Edition
postgres	PostgreSQL
salesforce	Salesforce
servicenow	ServiceNow
snowflake	Snowflake

Autonomous Database automatically configures and handles the secure connection to a target database and your connections are end-to-end encrypted. Oracle-managed heterogeneous connectivity is preconfigured with a wallet that contains most of the common trusted root and intermediate SSL certificates. Thus, NULL must be provided as the value for the ssl_server_cert_dn parameter.

To ensure security when using database links with Oracle-managed heterogeneous connectivity, the connection port is restricted and must have SSL/TLS enabled. You specify the target database port with the port parameter.

See Oracle-Managed Heterogeneous Connectivity Database Types and Ports for the list of supported non-Oracle database types.

The HETEROGENEOUS_CONNECTIVITY_INFO view provides information on supported Oracle Heterogeneous Connectivity types and shows a PL/SQL code sample for each supported type. See Access Heterogeneous Connectivity Information and Samples for more information.

When you specify the gateway_params parameter, for some db_type values, additional gateway_params parameters are supported:

db_type Value	Supported Gateway Parameters
google_analytics	When the db_type value is google_analytics the hostname is not used and you can provide value such as example.com. For db_type google_analytics, the credential must be an OAuth type credential using the params parameter with gcp_oauth2 values specified (client_id, client_secret, and refresh_token). See CREATE_CREDENTIAL Procedure for more information.
google_bigquery	When db_type is google_bigquery, the parameter project is valid. This parameter specifies the project name on google_bigquery and is required. When the db_type value is google_bigquery the hostname is not used and you can provide value such as example.com. For db_type google_bigquery, the credential must be an OAuth type credential using the params parameter with gcp_oauth2 values specified (client_id, client_secret, and refresh_token). See CREATE_CREDENTIAL Procedure for more information.
hive	When db_type is hive, the parameter http_path is valid. This parameter specifies the HttpPath value, if required, to connect to the Hive instance.

db_type Value	Supported Gateway Parameters
salesforce	When you use gateway_params parameter with db_type salesforce, you must supply the security_token option. The security token is a case-sensitive alphanumeric code. See Reset Your Security Token for more information. When you use gateway_params parameter with db_type salesforce, you must supply the correct hostname parameter. Salesforce provides two forms of URLs for the Salesforce service account: • *xxxx*.develop.lightning.force.com form of URL • *xxxx*my.salesforce.com form of URL as shown under the **View profile** tab. Oracle-Managed Heterogeneous Connectivity only supports the *xxxx*my.salesforce.com form of URL for the hostname parameter value.
servicenow	To connect to ServiceNow and get data you must supply the gateway parameters directory_name and file_name. These parameters specify a model file (REST config file) that maps the JSON response to the relational model. The model file specifies the endpoints, table mapping, and HTTP response code for processing the JSON response. See Model file syntax and Example Model file for more information. When you use gateway_params parameter with db_type servicenow there are two supported options: • Basic Authentication: you must supply the gateway_params parameter db_type with the value 'servicenow', and supply the directory_name and file_name parameters along with username/password type credentials. • OAuth 2.0 Authentication: you must supply the gateway_params parameter db_type with the value 'servicenow', and the directory_name, file_name, and token_uri parameters, along with OAuth type credentials. For OAuth 2.0 authentication with db_type servicenow, the credential must be an OAuth type credential using the params parameter with gcp_oauth2 values specified specified (client_id, client_secret, and refresh_token). See CREATE_CREDENTIAL Procedure for more information. The directory_name parameter specifies the directory with the ServiceNow REST config file. You could create this directory as follows: `create or replace directory servicenow_dir as 'SERVICENOW_DIR';` Obtain and download the ServiceNow REST config file to the specified directory. For example: `exec DBMS_CLOUD.get_object('servicenow_dir_cred', 'https://objectstorage.<...>/ servicenow.rest','SERVICENOW_DIR');` Set the file_name value to the name of the REST config file you downloaded, "servicenow.rest". Then you can use the ServiceNow REST config file with either basic authentication or OAuth2.0. See HETEROGENEOUS_CONNECTIVITY_INFO View for samples.

db_type Value	Supported Gateway Parameters
snowflake	When you use gateway_params parameter with db_type snowflake, use the Snowflake account identifier as the hostname parameter. In this case, the driver adds snowflakecomupting.com, so you do not pass this part of the hostname explicitly. To find your Snowflake account identifier, see Account Identifier Formats by Cloud Platform and Region. For example: for the Snowflake account: https://example-marketing_test_account.snowflakecomputing.com Set the hostname value to "example-marketing_test_account".

See CREATE_DATABASE_LINK Procedure for more information.

3. Use the database link to access data on the target database.

   For example:

```
SELECT count(*) FROM sales@AWSREDSHIFT_LINK
```

   The table name you specify when you use SELECT with Google BigQuery must be in quotes. For example:

```
SELECT count(*) FROM "sales"@GOOGLE_BIGQUERY_LINK
```

> **Note:**
>
> For the credentials you create in Step 1, the target database credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:
>
> ```
> BEGIN
>     DBMS_CLOUD.UPDATE_CREDENTIAL (
>        credential_name => 'AWS_REDSHIFT_LINK_CRED',
>        attribute => 'PASSWORD',
>        value => 'password');
> END;
> /
> ```
>
> Where *password* is the new password.
>
> After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

For additional information, see:

- Supported SQL Syntax and Functions in *Oracle Database Gateway for ODBC User's Guide*
- CREATE_DATABASE_LINK Procedure
- UPDATE_CREDENTIAL Procedure

- Create Database Links to Oracle MySQL on a Private Endpoint with Oracle-Managed Heterogeneous Connectivity
  Autonomous Database support for Oracle-managed heterogeneous connectivity makes it easy to create database links to Oracle MySQL Database Service on a private endpoint. When you use database links with Oracle-managed heterogeneous connectivity, Autonomous Database configures and sets up the connection to the Oracle MySQL Database Service.
- Oracle-Managed Heterogeneous Connectivity Database Types and Ports
  Shows the non-Oracle databases that you can connect to from Autonomous Database with Oracle-managed heterogeneous connectivity, and lists the supported port value for each database type. Also provides a link where you can see the supported database versions for each database type.
- Access Heterogeneous Connectivity Information and Samples
  Oracle Autonomous Database provides heterogeneous connectivity information for database links to non-Oracle databases.

Create Database Links to Oracle MySQL on a Private Endpoint with Oracle-Managed Heterogeneous Connectivity

Autonomous Database support for Oracle-managed heterogeneous connectivity makes it easy to create database links to Oracle MySQL Database Service on a private endpoint. When you use database links with Oracle-managed heterogeneous connectivity, Autonomous Database configures and sets up the connection to the Oracle MySQL Database Service.

> **Note:**
>
> Database links from an Autonomous Database to an Oracle MySQL Database Service that is on a private endpoint are only supported in commercial regions and US Government regions.
> This feature is enabled by default in all commercial regions.
>
> This feature is enabled by default in US Government regions for newly provisioned databases.
>
> For existing US Government databases on a private endpoint, if you want to create database links from an Autonomous Database to a target in a US Government region, please file a Service Request at Oracle Cloud Support and request to enable the private endpoint in government regions database linking feature.
>
> US Government regions include the following:
>
> - Oracle Cloud Infrastructure US Government Cloud with FedRAMP Authorization
> - Oracle Cloud Infrastructure US Federal Cloud with DISA Impact Level 5 Authorization

The following are prerequisites to use Oracle-managed heterogeneous connectivity with Oracle MySQL Database Service on a private endpoint:

- Create a DNS name using private DNS Zone pointing to private IP of your Oracle MySQL Database Service in your VCN. See Private DNS.
- Create an Autonomous Database with a Private Endpoint on same subnet.
- Configure the VCN for the Oracle MySQL Database Service to allow incoming connections on port 3306.

To create database links to a Oracle MySQL Database Service on a private endpoint using Oracle-managed heterogeneous connectivity, do the following:

1. On Autonomous Database create credentials to access the Oracle MySQL Database Service. The `username` and `password` you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the Oracle MySQL Database Service used within the database link.

   > **Note:**
   >
   > Supplying the `credential_name` parameter is required.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'MYSQL_LINK_CRED',
       username => 'NICK',
       password => 'password'
     );
   END;
   /
   ```

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

2. Create the database link to the Oracle MySQL Database Service using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

   For example, to create a database link:

   ```
   BEGIN
        DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
             db_link_name => 'MYSQL_LINK',
             hostname => 'mysql.example.com',
             port => '3306',
             service_name => 'mysql.example_service_name',
             ssl_server_cert_dn => NULL,
             credential_name => 'MYSQL_LINK_CRED',
             private_target => TRUE,
             gateway_params => JSON_OBJECT('db_type' value 'mysql'));
   END;
   /
   ```

3. Use the database link to access data on the target database.

   For example:

   ```
   SELECT count(*) FROM sales@MYSQL_LINK
   ```

> **Note:**
>
> For the credentials you create in Step 1, the target database credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:
>
> ```
> BEGIN
>     DBMS_CLOUD.UPDATE_CREDENTIAL (
>        credential_name => 'MYSQL_LINK_CRED',
>        attribute => 'PASSWORD',
>        value => 'password');
> END;
> /
> ```
>
> Where *password* is the new password.
>
> After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

For additional information, see:

- Supported SQL Syntax and Functions in *Oracle Database Gateway for ODBC User's Guide*
- CREATE_DATABASE_LINK Procedure
- UPDATE_CREDENTIAL Procedure

Oracle-Managed Heterogeneous Connectivity Database Types and Ports

Shows the non-Oracle databases that you can connect to from Autonomous Database with Oracle-managed heterogeneous connectivity, and lists the supported port value for each database type. Also provides a link where you can see the supported database versions for each database type.

> **Note:**
>
> Oracle uses Progress DataDirect connectors. The Database Support column provides links to the Progress website where you can find the supported database versions for each database type.

Database Type	db_type Value	Required Port	Database Support
Amazon Redshift	`awsredshift`	5439	Amazon Redshift Supported Versions
Azure SQL Microsoft SQL Server Azure Synapse Analytics	`azure`	1433	Azure SQL Supported Versions Azure Synapse Analytics Supported Versions
Apache Hive	`hive`	443	Hive Supported Versions
Google Analytics	`google_analytics`	443	Google Analytics Supported Versions

Database Type	db_type Value	Required Port	Database Support
Google BigQuery	`google_bigquery`	443	Google BigQuery Supported Versions
Apache Hive	`hive`	443	Hive Supported Versions
IBM Db2 11.5.6 or greater IBM Db2 11.5.5 or less	`db2`	25000 50000	IBM Db2 Supported Versions
MongoDB	`mongodb`	27017	MongoDB Supported Versions
MySQL	`mysql`	3306	MySQL Supported Versions
MySQL Community Edition	`mysql_community`	3306	
PostgreSQL	`postgres`	5432	PostgreSQL Supported Versions
Salesforce	`salesforce`	19937	Salesforce Supported Versions
ServiceNow	`servicenow`	443	ServiceNow Supported Versions
Snowflake	`snowflake`	443	Snowflake Supported Versions

Access Heterogeneous Connectivity Information and Samples

Oracle Autonomous Database provides heterogeneous connectivity information for database links to non-Oracle databases.

The `HETEROGENEOUS_CONNECTIVITY_INFO` view provides information on supported Oracle Heterogeneous Connectivity types and shows a PL/SQL code sample for each supported type.

For example:

```
SELECT * FROM HETEROGENEOUS_CONNECTIVITY_INFO WHERE DATABASE_TYPE = 'hive';
```

- HETEROGENEOUS_CONNECTIVITY_INFO View
  The `HETEROGENEOUS_CONNECTIVITY_INFO` view lists connectivity information and examples for connecting with PL/SQL using database links and Oracle Managed Heterogeneous Connectivity.

HETEROGENEOUS_CONNECTIVITY_INFO View

The `HETEROGENEOUS_CONNECTIVITY_INFO` view lists connectivity information and examples for connecting with PL/SQL using database links and Oracle Managed Heterogeneous Connectivity.

Column	Datatype	Description
`DATABASE_TYPE`	`VARCHAR2(32)`	Database type value used with `gateway_params` parameter.
`REQUIRED_PORT`	`NUMBER`	Supported port values for the database type.
`DESCRIPTION`	`CLOB`	Specifies a description for the `DATABASE_TYPE`.
`OPTIONAL_PARAMETERS`	`VARCHAR2(1024)`	Specifies the valid optional parameters for the `DATABASE_TYPE`.

Column	Datatype	Description
SAMPLE_USAGE	CLOB	Shows sample PL/SQL usage for the DATABASE_TYPE.

Create Database Links to Non-Oracle Databases with Customer-Managed Heterogeneous Connectivity

Use DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK to create database links from an Autonomous Database to an Oracle Database Gateway to access Non-Oracle databases.

Topics

- Prerequisites to Create Database Links to Non-Oracle Databases with Customer-Managed Heterogeneous Connectivity
  Use DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK to create database links from an Autonomous Database to an Oracle Database Gateway to access Non-Oracle databases.
- Create Database Links with Customer-Managed Heterogeneous Connectivity to Publicly Accessible Non-Oracle Databases
  Use DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK to create database links from an Autonomous Database instance on a public endpoint to an Oracle Database Gateway to access Non-Oracle databases.
- Create Database Links with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint
  You can create database links from an Autonomous Database to an Oracle Database Gateway to access Non-Oracle databases that are on a private endpoint.

Prerequisites to Create Database Links to Non-Oracle Databases with Customer-Managed Heterogeneous Connectivity

Use DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK to create database links from an Autonomous Database to an Oracle Database Gateway to access Non-Oracle databases.

An Oracle Database Gateway is a gateway that is designed for accessing a specific non-Oracle system. Using an Oracle Database Gateway, you can access data anywhere in a distributed database system without knowing either the location of the data or how it is stored. Using database links on Autonomous Database with Oracle Database Gateway supports heterogeneous environments and eliminates the need to customize your applications to access data from non-Oracle systems.

> **Note:**
>
> Before you create database links to a target gateway, do the following:
>
> 1. Configure the Oracle Database Gateway to access a non-Oracle database.
> 2. Configure Oracle Net Listener to handle incoming requests on the Oracle Database Gateway.
> 3. Create a self signed wallet on the Oracle Database Gateway.

For more information on Oracle Database Gateway, see Oracle Database Gateways in *Oracle Database Heterogeneous Connectivity User's Guide*. Also see the Installation and Configuration Guide and the Gateway User's Guide for the database you want to connect to. For example, for Oracle Database Gateway for SQL Server see:

- Installing and Configuring Oracle Database Gateway for SQL Server
- Introduction to the Oracle Database Gateway for SQL Server in *Oracle Database Gateway for SQL Server User's Guide*.
- Configure Oracle Net for the Gateway

Create Database Links with Customer-Managed Heterogeneous Connectivity to Publicly Accessible Non-Oracle Databases

Use `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to create database links from an Autonomous Database instance on a public endpoint to an Oracle Database Gateway to access Non-Oracle databases.

To use database links from an Autonomous Database instance on a public endpoint, the target gateway must be configured to use TCP/IP with SSL (TCPS) authentication. See Configuring Secure Sockets Layer Authentication for more information.

To create database links from an Autonomous Database instance on a public endpoint to a target gateway, do the following:

1. Copy the target gateway self signed wallet, for example, `cwallet.sso`, containing the certificates for the Oracle Database Gateway to Object Store.

   > **Note:**
   >
   > The wallet file, along with the Database user ID and password provide access to data available through the target gateway. Store wallet files in a secure location. Share wallet files only with authorized users.

2. Create credentials to access the Object Store where you store the `cwallet.sso`. See CREATE_CREDENTIAL Procedure for information about the username and password parameters for different object storage services.
3. Create a directory on Autonomous Database for the wallet file `cwallet.sso`.

   For example:

   ```
   CREATE DIRECTORY dblink_wallet_dir AS 'directory_path_of_your_choice';
   ```

   See Create Directory in Autonomous Database for information on creating directories.
4. Use `DBMS_CLOUD.GET_OBJECT` to upload the target gateway self signed wallet to the directory you created in the previous step, *DBLINK_WALLET_DIR*.

   For example:

   ```
   BEGIN
        DBMS_CLOUD.GET_OBJECT(
           credential_name => 'DEF_CRED_NAME',
           object_uri => 'https://objectstorage.us-
   phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/cwallet.sso',
           directory_name => 'DBLINK_WALLET_DIR');
   ```

```
END;
/
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.

The `credential_name` you use in this step is the credentials for the Object Store. In the next step you create the credentials to access the target gateway.

5. On Autonomous Database create credentials to access the target database. The `username` and `password` you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database used within the database link, (where the target database is accessed through the Oracle Database Gateway).

   Supplying the `credential_name` parameter is required.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DB_LINK_CRED',
       username => 'NICK',
       password => 'password'
     );
   END;
   /
   ```

   The characters in the `username` parameter must be all uppercase letters.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

6. Create the database link to the target gateway using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

   For example:

   ```
   BEGIN
        DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
             db_link_name =>       'SALESLINK',
             hostname =>           'example.com',
             port =>               '1522',
             service_name =>       'example_service_name',
             ssl_server_cert_dn => 'ssl_server_cert_dn',
             credential_name =>    'DB_LINK_CRED',
             directory_name =>     'DBLINK_WALLET_DIR',
             gateway_link =>        TRUE);
   END;
   /
   ```

   Users other than ADMIN require privileges to run `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

7. Use the database link you created to access data on the target gateway.

For example:

```
SELECT * FROM employees@SALESLINK;
```

For the credentials you create in Step 5, the Oracle Database Gateway credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:

```
BEGIN
    DBMS_CLOUD.UPDATE_CREDENTIAL (
       credential_name => 'DB_LINK_CRED',
       attribute => 'PASSWORD',
       value => 'password');
END;
/
```

Where *password* is the new password.

After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

See the following for an example that shows you how to create a database link to an Oracle Database Gateway to access a Microsoft SQL Server database:

How to Access Non-Oracle Databases from Autonomous Database using Oracle Database Gateway

For additional information, see:

- CREATE_DATABASE_LINK Procedure
- GET_OBJECT Procedure and Function
- UPDATE_CREDENTIAL Procedure

Create Database Links with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint

You can create database links from an Autonomous Database to an Oracle Database Gateway to access Non-Oracle databases that are on a private endpoint.

This section covers the steps for using database links to connect from Autonomous Database to a non-Oracle Database that through an Oracle Database Gateway, where the non-Oracle Database is on a private endpoint.

- Prerequisites to Create Database Links with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint
  Lists the prerequisites to create database links from an Autonomous Database with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases that are on a Private Endpoint.
- Create Database Links with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint (without a wallet)
  Create database links from an Autonomous Database to an Oracle Database Gateway to access Non-Oracle databases that are on a private endpoint without a wallet (TCP).

- Create Database Links with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint (with a Wallet)
  Create database links from an Autonomous Database to an Oracle Database Gateway to access Non-Oracle databases that are on a private endpoint (connecting with a wallet TCPS).

Prerequisites to Create Database Links with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint

Lists the prerequisites to create database links from an Autonomous Database with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases that are on a Private Endpoint.

To create a database link with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases that are on a Private Endpoint:

- The target database must be accessible from the source database's Oracle Cloud Infrastructure VCN. For example, you can connect to the target database when:
  - The target database is on a private endpoint.
  - Both the source database and the target database are in the same Oracle Cloud Infrastructure VCN.
  - The source database and the target database are in different Oracle Cloud Infrastructure VCNs that are paired.
  - For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a SCAN IP, or a SCAN hostname).
- The following ingress and egress rules must be defined for the private endpoint:
  - Define an egress rule in the source database's subnet security list or network security group such that the traffic over TCP is allowed to the target database's IP address and port number.
  - Define an ingress rule in the target database's subnet security list or network security group such that the traffic over TCP is allowed from the source database IP address to the destination port.

  See Configure Network Access with Private Endpoints for information on configuring private endpoints with ingress and egress rules.

> **Note:**
>
> When your Autonomous Database instance is configured with a private endpoint, set the `ROUTE_OUTBOUND_CONNECTIONS` database parameter to '`PRIVATE_ENDPOINT`' to specify that all outgoing database links are subject to the Autonomous Database instance private endpoint VCN's egress rules. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

Create Database Links with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint (without a wallet)

Create database links from an Autonomous Database to an Oracle Database Gateway to access Non-Oracle databases that are on a private endpoint without a wallet (TCP).

You can create database links from an Autonomous Database to an Oracle Database Gateway to access Non-Oracle databases that are on a private endpoint either with a wallet (TCPS), or without a wallet (TLS). This section describes creating a database link without a wallet.

To create database links from an Autonomous Database instance on a private endpoint to a target gateway, using database links and a TLS connection, do the following:

1. On Autonomous Database create credentials to access the target database. The `username` and `password` you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database used within the database link, (where the target database is accessed through the Oracle Database Gateway).

   Supplying the `credential_name` parameter is required.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DB_LINK_CRED',
       username => 'NICK',
       password => 'password'
     );
   END;
   /
   ```

   The characters in the `username` parameter must be all uppercase letters.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

2. Create the database link to the target gateway using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

   For example:

   ```
   BEGIN
        DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
             db_link_name =>       'SALESLINK',
             hostname =>           'example.com',
             port =>               '1522',
             service_name =>       'example_service_name',
             ssl_server_cert_dn => 'ssl_server_cert_dn',
             credential_name =>    'DB_LINK_CRED',
             directory_name =>      NULL,
             gateway_link =>        TRUE,
             private_target =>      TRUE,
             gateway_params =>      NULL);
   END;
   /
   ```

   The `ssl_server_cert_dn` parameter is optional if the connection is created as a TCP based database link (without a wallet).

   When the `directory_name` is `NULL`, the connection is created as a TCP based database link (without a wallet).

The `private_target` parameter must be set to `TRUE` when the target non-Oracle Database is on a private endpoint (that is, the database link accesses a hostname that needs to be resolved in a VCN DNS server). When `private_target` is `TRUE`, the `hostname` parameter must be a single hostname (on a private endpoint, using an IP address, a SCAN IP, or a SCAN hostname is not supported).

When `gateway_link` is `TRUE` and `gateway_params` is `NULL`, this specifies that the database link is to a customer-managed Oracle gateway.

Users other than ADMIN require privileges to run `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

3. Use the database link you created to access data on the target gateway.

   For example:

   ```
   SELECT * FROM employees@SALESLINK;
   ```

For the credentials you create in Step 1, the Oracle Database Gateway credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:

```
BEGIN
    DBMS_CLOUD.UPDATE_CREDENTIAL (
       credential_name => 'DB_LINK_CRED',
       attribute => 'PASSWORD',
       value => 'password');
END;
/
```

Where *password* is the new password.

After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

For additional information, see:

- CREATE_DATABASE_LINK Procedure
- UPDATE_CREDENTIAL Procedure

Create Database Links with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint (with a Wallet)

Create database links from an Autonomous Database to an Oracle Database Gateway to access Non-Oracle databases that are on a private endpoint (connecting with a wallet TCPS).

You can create database links from an Autonomous Database to an Oracle Database Gateway to access Non-Oracle databases that are on a private endpoint either with a wallet (TCPS), or without a wallet (TLS). This section describes creating a database link with a wallet.

To create database links from an Autonomous Database instance on a private endpoint to a target gateway, connecting with a wallet, do the following:

1. Copy the target gateway self signed wallet, for example, `cwallet.sso`, containing the certificates for the Oracle Database Gateway to Object Store.

   The wallet file, along with the Database user ID and password provide access to data available through the target gateway. Store wallet files in a secure location. Share wallet files only with authorized users.

2. Create credentials to access the Object Store where you store the `cwallet.sso`. See CREATE_CREDENTIAL Procedure for information about the username and password parameters for different object storage services.
3. Create a directory on Autonomous Database for the wallet file `cwallet.sso`.

   For example:

   ```
   CREATE DIRECTORY dblink_wallet_dir AS 'directory_path_of_your_choice';
   ```

   See Create Directory in Autonomous Database for information on creating directories.
4. Use `DBMS_CLOUD.GET_OBJECT` to upload the target gateway self signed wallet to the directory you created in the previous step, *DBLINK_WALLET_DIR*.

   For example:

   ```
   BEGIN
        DBMS_CLOUD.GET_OBJECT(
           credential_name => 'DEF_CRED_NAME',
           object_uri => 'https://objectstorage.us-
   phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/cwallet.sso',
           directory_name => 'DBLINK_WALLET_DIR');
   END;
   /
   ```

   In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.

   > **Note:**
   >
   > The `credential_name` you use in this step is the credentials for the Object Store. In the next step you create the credentials to access the target gateway.

5. On Autonomous Database create credentials to access the target database. The `username` and `password` you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database used within the database link, (where the target database is accessed through the Oracle Database Gateway).

   Supplying the `credential_name` parameter is required.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DB_LINK_CRED',
       username => 'NICK',
       password => 'password'
     );
   END;
   /
   ```

   The characters in the `username` parameter must be all uppercase letters.

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

6. Create the database link to the target gateway using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

   For example:

```
BEGIN
     DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
          db_link_name =>       'SALESLINK',
          hostname =>           'example.com',
          port =>               '1522',
          service_name =>       'example_service_name',
          ssl_server_cert_dn => 'ssl_server_cert_dn',
          credential_name =>    'DB_LINK_CRED',
          directory_name =>     'DBLINK_WALLET_DIR',
          gateway_link =>        TRUE,
          private_target =>      TRUE,
          gateway_params =>      NULL);
END;
/
```

   If `directory_name` is not `NULL`, a TCPS-based database link is created.

   The `private_target` parameter must be set to `TRUE` when the target non-Oracle Database is on a private endpoint (that is, the database link accesses a hostname that needs to be resolved in a VCN DNS server). When `private_target` is `TRUE`, the `hostname` parameter must be a single hostname (on a private endpoint, using an IP address, a SCAN IP, or a SCAN hostname is not supported).

   When `gateway_link` is `TRUE` and `gateway_params` is `NULL`, this specifies that the database link is to a customer-managed Oracle gateway.

   Users other than ADMIN require privileges to run `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

7. Use the database link you created to access data on the target gateway.

   For example:

```
SELECT * FROM employees@SALESLINK;
```

For the credentials you create in Step 5, the Oracle Database Gateway credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:

```
BEGIN
    DBMS_CLOUD.UPDATE_CREDENTIAL (
       credential_name => 'DB_LINK_CRED',
       attribute => 'PASSWORD',
       value => 'password');
END;
/
```

Where *password* is the new password.

After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

For additional information, see:

- CREATE_DATABASE_LINK Procedure
- GET_OBJECT Procedure and Function
- UPDATE_CREDENTIAL Procedure

Create Database Links from Other Databases to Autonomous Database

You can create database links to an Autonomous Database from an Oracle database that is not an Autonomous Database. For example from an on-premise Oracle database to an Autonomous Database

- Supported Oracle Database versions for database links where the source is an Oracle Database and the target is an Autonomous Database instance are: 19.2 (or later), 21 (base release or later).
- If you are using database links between Autonomous Database and other Oracle Databases, you might need to apply Patch 33843368 on the Oracle Database that is not an Autonomous Database. This applies to cases where the Autonomous Database instance is either the source or the target of the database link.

  See My Oracle Support Knowledge Base: Patch Requirement For Database Links Between ADB-S And Other Oracle Databases(Doc ID 2874244.1) for further details.

To create database links to an Autonomous Database do the following:

1. Download your Autonomous Database wallet. See Download Client Credentials (Wallets) for more information.
2. Upload the wallet to the database instance where you want to create the link to the Autonomous Database.
3. Unzip the Autonomous Database wallet:

> **Note:**
>
> The wallet file, along with the Database user ID and password provide access to data in your Autonomous Database. Store wallet files in a secure location. Share wallet files only with authorized users.

```
[oracle@sys1 ~]$ cd/u01/targetwallet
[oracle@sys1 targetwallet]$ unzip Wallet_name1.zip
        Archive: Wallet_name1.zip
          inflating: cwallet.sso
          inflating: tnsnames.ora
          inflating: truststore.jks
          inflating: ojdbc.properties
          inflating: sqlnet.ora
          inflating: ewallet.p12
          inflating: keystore.jks
```

4. Set `GLOBAL_NAMES` to `FALSE`.

```
SQL> ALTER SYSTEM SET GLOBAL_NAMES = FALSE;

System altered.

SQL> SHOW PARAMETER GLOBAL_NAMES
NAME                      TYPE        VALUE
------------------------  ----------- -----------
global_names              boolean     FALSE
```

   Set `GLOBAL_NAMES` to `FALSE` to use a database link name without checking that the name is different than the remote database name.When `GLOBAL_NAMES`, is set to `TRUE`, the database requires the database link to have the same name as the database to which it connects. See GLOBAL_NAMES for more information.

5. Create the database link to the target Autonomous Database. Note that the `security` path includes `my_wallet_directory`; the path where you unzip the Autonomous Database wallet.

```
CREATE DATABASE LINK ADBLINK
   CONNECT TO NAME1 IDENTIFIED BY ************
   USING
'(description=(retry_count=20)(retry_delay=3)
     (address=(protocol=tcps)(port=1522)
(host=example1.oraclecloud.com))
     (connect_data=(service_name=example2_high.adb.oraclecloud.com))
     (security=(my_wallet_directory=/u01/targetwallet)
        (ssl_server_dn_match=true)))';

Database link created.
```

6. Use the database link you created to access data on the target database (your Autonomous Database instance in this case):

   For example:

```
SELECT * FROM employees@ADBLINK;
```

To list the database links, use the `ALL_DB_LINKS` view. See ALL_DB_LINKS for more information.

For additional information, see:

- See CREATE DATABASE LINK for details on the procedure.
- See Create Database Links from Autonomous Database to a Publicly Accessible Autonomous Database with a Wallet (mTLS)

Drop Database Links

After you create a database link you can drop the database link.

- Drop a database link to a target database using `DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK`.

For example:

```
BEGIN
     DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK(
          db_link_name => 'SALESLINK' );
END;
/
```

See DROP_DATABASE_LINK Procedure for detailed information about the procedure.

Call Web Services from Autonomous Database

Describes options for calling Web Services from Autonomous Database.

There are a number of options for calling Web Services from Autonomous Database, including the following:

- Use `DBMS_CLOUD` REST APIs: The `DBMS_CLOUD.SEND_REQUEST` function begins an HTTP request, gets the response, and ends the response. This function provides a workflow for sending a cloud REST API request with arguments and the function returns a response code and payload. See SEND_REQUEST Function and Procedure for more information.
- Use Oracle APEX: You can interact with both SOAP and RESTful style web services from APEX in your Autonomous Database instance. See Use Web Services with Oracle APEX for more information.
- Use `UTL_HTTP` to submit a request to a public site: See Submit an HTTP Request to a Public Host for more information.
- Use `UTL_HTTP` to submit a request to a private site: See Submit an HTTP Request to a Private Host for more information.

See PL/SQL Packages Notes for Autonomous Database for information on restrictions for `UTL_HTTP` on Autonomous Database.

Topics

- Submit an HTTP Request to a Public Host
  Provides details for using `UTL_HTTP` to submit an HTTP request on a public host.
- Submit an HTTP Request to a Private Host
  Describes the steps to use `UTL_HTTP` to submit an HTTP request on a private host.
- Submit an HTTP Request to Private Site with a Proxy
  When your Autonomous Database instance is on a private endpoint you can use a proxy to submit HTTP requests with `UTL_HTTP`.
- Use Credential Objects to Set HTTP Authentication
  Describes how to pass a credential objects to `UTL_HTTP.SET_CREDENTIAL` procedure.

Submit an HTTP Request to a Public Host

Provides details for using `UTL_HTTP` to submit an HTTP request on a public host.

For example, to submit an HTTP request for a public host www.example.com, create an Access Control List for the host:

```
BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'www.example.com',
         ace =>  xs$ace_type( privilege_list => xs$name_list('http'),
                              principal_name => 'ADMIN',
                              principal_type => xs_acl.ptype_db));
END;
```

Then submit the HTTP request:

```
SELECT UTL_HTTP.REQUEST(url => 'https://www.example.com/') FROM dual;
```

> **Note:**
>
> If your Autonomous Database instance is on a private endpoint and you want your `UTL_HTTP` calls to public hosts to be subject to your private endpoint VCN's egress rules, set the `ROUTE_OUTBOUND_CONNECTIONS` database property to `PRIVATE_ENDPOINT`. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

See PL/SQL Packages Notes for Autonomous Database for information on restrictions for `UTL_HTTP` on Autonomous Database.

Submit an HTTP Request to a Private Host

Describes the steps to use `UTL_HTTP` to submit an HTTP request on a private host.

To submit a request to a target host on a private endpoint, the target host must be accessible from the source database's Oracle Cloud Infrastructure VCN. For example, you can connect to the target host when:

- Both the source database and the target host are in the same Oracle Cloud Infrastructure VCN.
- The source database and the target host are in different Oracle Cloud Infrastructure VCNs that are paired.
- The target host is an on-premises network that is connected to the source database's Oracle Cloud Infrastructure VCN using FastConnect or VPN.

> **Note:**
>
> Making `UTL_HTTP` calls to a private host is only supported in commercial regions and US Government regions.
> This feature is enabled by default in all commercial regions.
>
> This feature is enabled by default in US Government regions for newly provisioned databases.
>
> For existing US Government databases on a private endpoint, if you want to make `UTL_HTTP` from an Autonomous Database to a target in a US Government region, please file a Service Request at Oracle Cloud Support and request to enable the private endpoint in government regions database linking feature.
>
> US Government regions include the following:
>
> - Oracle Cloud Infrastructure US Government Cloud with FedRAMP Authorization
> - Oracle Cloud Infrastructure US Federal Cloud with DISA Impact Level 5 Authorization

To make a `UTL_HTTP` request to a target on a private endpoint:

1. Create an Access Control List for the host.

   For example:

```
BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'www.example.com',
         ace => xs$ace_type( privilege_list => xs$name_list('http'),
                             principal_name => 'ADMIN',
                             principal_type => xs_acl.ptype_db),
                             private_target => TRUE);
END;
/
```

   As shown in this example, when you create an Access Control List for the host specify the `private_target` parameter with the value `TRUE`.

   > **Note:**
   >
   > If you set `ROUTE_OUTBOUND_CONNECTIONS` to `PRIVATE_ENDPOINT`, setting the `private_target` parameter to `TRUE` is not required in this API. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

2. Submit the HTTP request:

```
SELECT UTL_HTTP.REQUEST(
                url => 'https://www.example.com/',
                https_host => 'www.example.com')
             FROM dual;
```

See PL/SQL Packages Notes for Autonomous Database for information on restrictions for `UTL_HTTP` on Autonomous Database.

Submit an HTTP Request to Private Site with a Proxy

When your Autonomous Database instance is on a private endpoint you can use a proxy to submit HTTP requests with `UTL_HTTP`.

When your Autonomous Database instance is on a private endpoint, to use `UTL_HTTP` with a target proxy the target proxy must be accessible from the source database's Oracle Cloud Infrastructure VCN.

For example, you can connect using a proxy when:

- Both the source database and the proxy server are in the same Oracle Cloud Infrastructure VCN.
- The source database and the proxy server are in different Oracle Cloud Infrastructure VCNs that are paired.
- The proxy server is an on-premises network that is connected to the source database's Oracle Cloud Infrastructure VCN using FastConnect or VPN.

> **Note:**
>
> Making `UTL_HTTP` calls to a proxy server on a private endpoint is only supported in commercial regions and US Government regions.
> This feature is enabled by default in all commercial regions.
>
> This feature is enabled by default in US Government regions for newly provisioned databases.
>
> For existing US Government databases on a private endpoint, if you want to make `UTL_HTTP` from an Autonomous Database to a target in a US Government region, please file a Service Request at Oracle Cloud Support and request to enable the private endpoint in government regions database linking feature.
>
> US Government regions include the following:
>
> - Oracle Cloud Infrastructure US Government Cloud with FedRAMP Authorization
> - Oracle Cloud Infrastructure US Federal Cloud with DISA Impact Level 5 Authorization

To use a proxy server with `UTL_HTTP`:

1. Set the HTTP_PROXY ACL on the proxy server.

   For example:

```
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
       host =>'www-proxy-example.com',
       ace  => xs$ace_type(privilege_list => xs$name_list('HTTP_PROXY'),
                           principal_name => 'APPUSER1',
                           principal_type => xs_acl.ptype_db),
                           private_target => TRUE);
```

```
END;
/
```

As shown in this example, when you create an Access Control List for the proxy server specify the `private_target` parameter with the value `TRUE`.

> **Note:**
>
> If you set `ROUTE_OUTBOUND_CONNECTIONS` to `PRIVATE_ENDPOINT`, setting the `private_target` parameter to `TRUE` is not required in this API. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

2. Set the HTTP ACL on the remote Web Server.

   For example:

```
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
        host =>'example.com',
        ace => xs$ace_type( privilege_list => xs$name_list('HTTP'),
                            principal_name => 'APPUSER1',
                            principal_type => xs_acl.ptype_db)
END;
/
```

3. Set the wallet and the proxy for `UTL_HTTP`.

   For example:

```
BEGIN
   UTL_HTTP.SET_WALLET('');
   UTL_HTTP.SET_PROXY('www-proxy-example:80');
END;
/
```

4. Submit an HTTP request:

```
SELECT UTL_HTTP.REQUEST(
                    url         => 'https://www.example.com/',
                    https_host  => 'www.example.com')
             FROM dual;
```

Notes for setting a proxy server with `UTL_HTTP.SET_PROXY`:

- `DBMS_CLOUD` requests do not honor the proxy server you set with `UTL_HTTP.SET_PROXY`. This includes `DBMS_CLOUD.SEND_REQUEST` and all object storage access for `DBMS_CLOUD` external tables that you define with `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`, `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`, or `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE`.
- `APEX_WEB_REQUEST` requests do not honor the proxy server you set with `UTL_HTTP.SET_PROXY`.

See PL/SQL Packages Notes for Autonomous Database for information on restrictions for `UTL_HTTP` on Autonomous Database.

Use Credential Objects to Set HTTP Authentication

Describes how to pass a credential objects to `UTL_HTTP.SET_CREDENTIAL` procedure.

The `UTL_HTTP.SET_CREDENTIAL` procedure sets HTTP authentication information in the HTTP request header. The Web server needs this information to authorize the request.

The `UTL_HTTP.SET_CREDENTIAL` procedure enables you to pass credential objects to set HTTP authentication. Credential objects are schema objects, hence they can be accessed only by privileged users and enable you to configure schema-level privileges to access control the credentials. Passing credential objects is an appropriate and secure way to store and manage username/password/keys to be used for authentication.

The `UTL_HTTP.SET_CREDENTIAL` procedure is a secure and convenient alternative to `UTL_HTTP.SET_AUTHENTICATION` procedure.

Example

```
...
UTL_HTTP.SET_AUTHENTICATION (l_http_request, 'web_app_user', 'xxxxxxxxxxx');
...
```

As shown in the example above, when you invoke `SET_AUTHENTICATION` procedure, you must pass the username/password in clear text as part of PL/SQL formal parameters. You might need to embed the username/password into various PL/SQL automation or cron scripts. Passing clear text passwords is a compliance issue that is addressed in `UTL_HTTP.SET_CREDENTIAL` procedure.

See SET_AUTHENTICATION Procedure and SET_AUTHENTICATION_FROM_WALLET Procedure for more information.

`UTL_HTTP.SET_CREDENTIAL` Syntax

```
UTL_HTTP.SET_CREDENTIAL (
    r          IN OUT NOCOPY req,
    credential IN VARCHAR2,
    scheme     IN VARCHAR2 DEFAULT 'Basic',
    for_proxy  IN BOOLEAN  DEFAULT FALSE);
```

Example to pass a credential object in the `SET_CREDENTIAL` procedure:

- Create a credential object:

  ```
  BEGIN DBMS_CLOUD.CREATE_CREDENTIAL (
      credential_name => 'HTTP_CRED',
      username        => 'web_app_user',
      password        => '<password>' );
  END;
  ```

  This creates a credential object which creates a stored username/password pair.

  See CREATE_CREDENTIAL Procedure for more information.

  See Specifying Scheduler Job Credentials for more information.

- Invoke `UTL_HTTP.SET_CREDENTIAL` procedure:

```
DECLARE
      l_http_request  UTL_HTTP.REQ;
    BEGIN
      l_http_request := UTL_HTTP.BEGIN_REQUEST('https://www.example.com/v1/
dwcsdev/NAME/dwcs_small_xt1.csv');
      UTL_HTTP.SET_CREDENTIAL (l_http_request, 'HTTP_CRED','BASIC');
      ...
END;
```

  This example first creates a request by invoking the `BEGIN_REQUEST` procedure and sets HTTP authentication information in the HTTP request header by invoking the `SET_CREDENTIAL` procedure. The Web server needs this information to authorize the request. The value `l_http_request` is the HTTP request, `HTTP_CRED` is the credentials name and `BASIC` is the HTTP authentication scheme.

See UTL_HTTP for more information.

See PL/SQL Packages Notes for Autonomous Database for information on restrictions for `UTL_HTTP` on Autonomous Database.

Move Files

Describes how to create directories and copy files to Autonomous Database.

- Creating and Managing Directories on Autonomous Database
- Copy Files Between Object Store and a Directory in Autonomous Database
  Use the procedure `DBMS_CLOUD.PUT_OBJECT` to copy a file from a directory to Object Store. Use the procedure `DBMS_CLOUD.GET_OBJECT` to copy a file from Object Store to a directory.
- Bulk Operations for Files in the Cloud
  The PL/SQL package `DBMS_CLOUD` offers parallel execution support for bulk file upload, download, copy, and transfer activities, which streamlines the user experience and delivers optimal performance for bulk file operations.

Creating and Managing Directories on Autonomous Database

Autonomous Database includes a predefined `data_pump_dir` directory in the database where you can place files. You can also create directories, drop directories, and attach network file systems.

For example you can use the `data_pump_dir` directory for Data Pump import and export operations. To create additional directories use the database `CREATE DIRECTORY` command. Use the database `DROP DIRECTORY` command to drop directories, and use `DBMS_CLOUD.LIST_FILES` to list the contents of a directory. You can attach a Network File System to a directory location in your Autonomous Database with the `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` procedure.

- Create Directory in Autonomous Database
  To create directories use the database `CREATE DIRECTORY` command. Using `CREATE DIRECTORY` you specify the path as a relative path for the new directory.
- Drop Directory in Autonomous Database
  Use the database `DROP DIRECTORY` command to drop a directory object.

- List Contents of Directory in Autonomous Database
  Use the function `DBMS_CLOUD.LIST_FILES` to list the contents of a directory.
- Access Network File System from Autonomous Database
  You can attach a Network File System to a directory location in your Autonomous Database.
- Load Data from Directories in Autonomous Database
  As an alternative to an object store location URI, you can specify a directory with `DBMS_CLOUD` procedures to load or unload data from files in a local directory, including directories created on attached network file systems.

Create Directory in Autonomous Database

To create directories use the database `CREATE DIRECTORY` command. Using `CREATE DIRECTORY` you specify the path as a relative path for the new directory.

`CREATE DIRECTORY` creates the database directory object and also creates the file system directory if it does not already exist. If the file system directory exists then `CREATE DIRECTORY` only creates the database directory object. For example, the following command creates the database directory named `staging` and creates the file system directory `stage`:

```
CREATE DIRECTORY staging AS 'stage';
```

You can also create subdirectories. For example, the following command creates the database directory object `sales_staging` and the file system directory `stage/sales`:

```
CREATE DIRECTORY sales_staging AS 'stage/sales';
```

When you create subdirectories you do not have to create the initial file system directory. For example, in the previous example if the directory `stage` does not exist then the `CREATE DIRECTORY` command creates both directories `stage` and `stage/sales`.

To add a directory, you must have the `CREATE ANY DIRECTORY` system privilege. The ADMIN user is granted the `CREATE ANY DIRECTORY` system privilege. The ADMIN user can grant `CREATE ANY DIRECTORY` system privilege to other users.

See CREATE DIRECTORY for more information.

Notes:

- `CREATE DIRECTORY` creates the database directory object in the database and also creates the file system directory. For example the directory path could be:

  ```
  /u03/dbfs/7C149E35BB1000A45FD/data/stage
  ```

- You can create a directory in the root file system to see all the files with the following commands:

  ```
  CREATE OR REPLACE DIRECTORY ROOT_DIR AS '';
  ```

  After you create the `ROOT_DIR` directory, use the following command to list all files:

  ```
  SELECT * FROM DBMS_CLOUD.list_files('ROOT_DIR');
  ```

  To run `DBMS_CLOUD.LIST_FILES` with a user other than ADMIN you need to grant read privileges on the directory to that user. See LIST_FILES Function for more information.
- Space in the file system allocated for the directories you create and their contents is part of your storage allocation. See Database Dashboard Overview to view the total space allocated.

Drop Directory in Autonomous Database

Use the database `DROP DIRECTORY` command to drop a directory object.

For example, the following command drops the database directory object `staging`:

```
DROP DIRECTORY staging;
```

The `DROP DIRECTORY` command does not delete files in the directory. If you want to delete the directory and the files in the directory, first use the procedure `DBMS_CLOUD.DELETE_FILE` to delete the files. See DELETE_FILE Procedure for more information.

To drop a directory, you must have the `DROP ANY DIRECTORY` system privilege. The ADMIN user is granted the `DROP ANY DIRECTORY` system privilege. The ADMIN user can grant `DROP ANY DIRECTORY` system privilege to other users.

See DROP DIRECTORY for more information.

Notes:

- You are not allowed to drop the predefined directories: `data_pump_dir` or `sql_tcb_dir`
- If you just want to drop the directory and you do not remove the files in the directory, after you drop the directory you can view all the files in the file system, including any files that were in the directory you dropped, as follows:

  ```
  CREATE OR REPLACE DIRECTORY ROOT_DIR AS '';
  ```

  Then list the contents of `ROOT_DIR` with the following command:

  ```
  SELECT * FROM DBMS_CLOUD.list_files('ROOT_DIR');
  ```

  To run `DBMS_CLOUD.LIST_FILES` with a user other than ADMIN you need to grant read privileges on the directory to that user. See LIST_FILES Function for more information.
- The `DROP DIRECTORY` command does not remove the underlying file system directory. Autonomous Database manages the underlying file system directory; users do not remove the file system directory.

List Contents of Directory in Autonomous Database

Use the function `DBMS_CLOUD.LIST_FILES` to list the contents of a directory.

For example, to list the contents of the `staging` directory, run the following query:

```
SELECT * FROM DBMS_CLOUD.LIST_FILES('STAGING');
```

To run `DBMS_CLOUD.LIST_FILES` with a user other than ADMIN you need to grant read privileges on the directory to that user. See LIST_FILES Function for more information.

Access Network File System from Autonomous Database

You can attach a Network File System to a directory location in your Autonomous Database.

This allows you to load data from Oracle Cloud Infrastructure File Storage in your Virtual Cloud Network (VCN) or from any other Network File System in on-premises data centers. Depending on the version of the Network File System you want to access, both NFSv3 and NFSv4 are supported.

Supporting Network File System allows you to do the following:

- Connect to an Autonomous Database instance from a legacy application and use the file system to load and unload data.
- Analyze data from different sources in an Autonomous Database.
- Secure access to data in an Autonomous Database from the file systems in an on-premises data center or Private Virtual Cloud Networks (VCNs).

Topics

- Attach Network File System to Autonomous Database
  Use `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` to attach a file system to a directory in your Autonomous Database.
- Detach Network File System from Autonomous Database
  Use the `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` procedure to detach a file system from a directory in your Autonomous Database.
- Example: Set Up an NFSv4 Server on Oracle Cloud Compute
  Provides an example for setting up an NSFv4 server for use with Autonomous Database.
- DBA_CLOUD_FILE_SYSTEMS View
  The `DBA_CLOUD_FILE_SYSTEMS` view lists information about the network file system attached to a directory location in the database.

Attach Network File System to Autonomous Database

Use `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` to attach a file system to a directory in your Autonomous Database.

With an attached file system you can load data from any of the following:

- Oracle Cloud Infrastructure File Storage in your Virtual Cloud Network (VCN).

  See How to Attach a File System to your Autonomous Database for details on setting up Oracle Cloud Infrastructure File Storage with Autonomous Database.
- From a Network File System in an on-premises data center. Depending on the version of the Network File System you want to access, both NFSv3 and NFSv4 are supported.

  See Example: Set Up an NFSv4 Server on Oracle Cloud Compute for a configuration example with an NFSv4 Network File System.

> **Note:**
>
> The`DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` procedure can only attach a private File Storage Service when the Autonomous Database instance is on a private endpoint.

To access data in an Autonomous Database from the file systems in an on-premises data center you must set up FastConnect or a Site-to-Site VPN to connect to the on-premises data center. See FastConnect and Site-to-Site VPN for more information.

1. Create a directory or use an existing directory to attach a Network File System in your Autonomous Database. You must have `WRITE` privilege on the directory object on your Autonomous Database instance to attach a file system to a directory location in the database.

   For example, the following command creates the database directory named `FSS_DIR` and creates the file system directory `fss`:

   ```
   CREATE DIRECTORY FSS_DIR AS 'fss';
   ```

   See Create Directory in Autonomous Database for more information.

2. Run `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` to attach a file system to a directory in your Autonomous Database. To run this procedure, you must be logged in as the ADMIN user or have `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

   - By default `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` uses NFSv3:

     ```
     BEGIN
        DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM(
         file_system_name      => 'FSS',
         file_system_location  => 'myhost.sub000445.myvcn.oraclevcn.com:/
     results',
         directory_name        => 'FSS_DIR',
         description           => 'Source NFS for sales data'
     );
     END;
     /
     ```

     Optionally you can use the `params` parameter and specify the `nfs_version` with value 3 to specify NFSv3.

   - To use NFSv4, include the `params` parameter with `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` and specify the `nfs_version` with value 4 to specify NFSv4:

     ```
     BEGIN
        DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM(
         file_system_name      => 'FSS',
         file_system_location  => 'myhost.sub000445.myvcn.oraclevcn.com:/
     results',
         directory_name        => 'FSS_DIR',
         description           => 'Source NFS for sales data',
         params                => JSON_OBJECT('nfs_version' value 4)
     );
     END;
     /
     ```

   These examples attach the network file system specified in the `file_system_name` parameter to the Autonomous Database.

   The `file_system_location` parameter specifies the location of the file system. The value you supply with `file_system_location` consists of a Fully Qualified Domain Name (FQDN) and a file path in the form: *FQDN*:*file_path*.

   For example:

   - FQDN: `myhost.sub000445.myvcn.oraclevcn.com`

     For Oracle Cloud Infrastructure File Storage set the FQDN in **Show Advanced Options** when you create a file system. See Creating File Systems for more information.

   - File Path: `/results`

   The `directory_name` parameter specifies the directory name in the Autonomous Database where you want to attach the file system. This is the directory you created in Step 1, or another directory you previously created.

   The `description` parameter specifies the description for the task.

The `params` parameter is a JSON value that specifies an additional attribute `nfs_version`, whose value can be either 3 or 4 (NFSv3 or NFSv4).

After you attach a file system you can query the `DBA_CLOUD_FILE_SYSTEMS` view to retrieve information about the attached file system.

For example:

```
SELECT file_system_name, file_system_location,
directory_path
   FROM dba_cloud_file_systems
   WHERE file_system_name = 'FSS';
```

This query returns details for the `FSS` file system name.

See DBA_CLOUD_FILE_SYSTEMS View for more information.

With an attached file system you can read and write to files on an attached file system using any PL/SQL API that accepts a directory name. For example, you can use any of the following methods to work with an attached NFS directory:

- The `UTL_FILE` package.
- Data Pump Export and Import utilities.
- The `DBMS_CLOUD` APIs that work with directories such as `DBMS_CLOUD.LIST_FILES` and `DBMS_CLOUD.PUT_OBJECT`.

Example showing a write a file on an attached file system using `UTL_FILE`:

```
DECLARE
  l_file         UTL_FILE.FILE_TYPE;
  l_location     VARCHAR2(100) := 'FSS_DIR';
  l_filename     VARCHAR2(100) := 'test.csv';
BEGIN
  -- Open the file.
  l_file := UTL_FILE.FOPEN(l_location, l_filename, 'w');

  UTL_FILE.PUT(l_file, 'Scott, male, 1000');

  -- Close the file.
  UTL_FILE.FCLOSE(l_file);
END;
/
```

Example showing a read a file on an attached file system using `UTL_FILE`:

```
DECLARE
  l_file         UTL_FILE.FILE_TYPE;
  l_location     VARCHAR2(100) := 'FSS_DIR';
  l_filename     VARCHAR2(100) := 'test.csv';
  l_text         VARCHAR2(32767);
BEGIN
  -- Open the file.
  l_file := UTL_FILE.FOPEN(l_location, l_filename, 'r');

  UTL_FILE.GET_LINE(l_file, l_text, 32767);
```

```
  -- Close the file.
  UTL_FILE.FCLOSE(l_file);
END;
/
```

Example showing list files on an attached file system using `DBMS_CLOUD.LIST_FILES`:

```
SELECT object_name FROM DBMS_CLOUD.LIST_FILES('FSS_DIR');
```

Notes for using `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM`:

- Oracle Cloud Infrastructure File Storage uses NFSv3 to share. See Overview of File Storage for more information.
- If you attach to non-Oracle Cloud Infrastructure File Storage systems, the procedure supports NFSv3 and NFSv4.
- If you have an attached NFS server that uses NFSv3 and the NFS version is updated to NFSv4 in the NFS server, you must run `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` and then `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` (using the `params` parameter with `nfs_version` set to 4). This attaches NFS with the matching protocol so that Autonomous Database can access the NFSv4 Server. Without detaching and then reattaching, the NFS server will be inaccessible and you may see an error such as: `"Protocol not supported"`.

See the following for more information:

- UTL_FILE
- LIST_FILES Function
- OCI File Storage Service
- ATTACH_FILE_SYSTEM Procedure

Detach Network File System from Autonomous Database

Use the `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` procedure to detach a file system from a directory in your Autonomous Database.

> **Note:**
>
> The `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` procedure can only detach a private File Storage Service from databases that are on private endpoints.

You must have `WRITE` privilege on the directory object to detach a file system from a directory location.

Run `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` procedure to detach a file system from a directory location in your Autonomous Database. To run this procedure, you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

For example:

```
BEGIN
   DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM (
    file_system_name      => 'FSS'
```

```
  );
END;
/
```

This example detaches the network file system specified in the `file_system_name` parameter from the Autonomous Database. You must provide a value for this parameter.

The information about this file system is removed from the `DBA_CLOUD_FILE_SYSTEMS` view.

See the following for more information:

- OCI File Storage Service
- Creating and Managing Directories on Autonomous Database
- DETACH_FILE_SYSTEM Procedure

Example: Set Up an NFSv4 Server on Oracle Cloud Compute

Provides an example for setting up an NSFv4 server for use with Autonomous Database.

1. Set up a private endpoint for the Autonomous Database instance.

   See Configuring Network Access with Private Endpoints for more information.

   The following ingress and egress rules need to be set for your VCN's security list so that Autonomous Database and the NFSv4 server can talk to each other

   - Stateful ingress from ALL ports in source CIDR block to TCP port 2049.
   - Stateful egress from TCP ALL ports to port 2049 in destination CIDR block.

2. Set up the NFS server on an Oracle Cloud VM with Oracle Linux 8 in the private subnet, which can connect to the Autonomous Database instance.

```
Compute Instance Imeage: Oracle-Linux-8.8-2023.09.26-0
$ sudo su -
$ mkdir /exports
$ chown nobody /exports
$ chgrp nobody /exports

# If the VM is using Linux 7, "data" directory may need having the
privilege 777 so that ADB can have access to NFS.
uname -a
chmod 777 /exports/data

$ mkdir /exports/data
$ chown nobody /exports/data
$ chgrp nobody /exports/data

# Either the private IP or the private FQDN can be used in "/etc/exports".
# Both can be found in the information of the autonomous database on the
OCI console.
$ tee -a /etc/exports <<'EOF'
  /exports/data *(rw,insecure)
  /exports/data example.adb.us-phoenix-1.oraclecloud.com(rw,insecure)
  EOF

$ systemctl start nfs-server
$ systemctl enable nfs-server
```

```
# Configure the firewall to allow NFS connections. "public" is the default
zone on Oracle Cloud VM.
$ firewall-cmd --get-default-zone
public

$ firewall-cmd --zone=public --add-service=nfs
$ firewall-cmd --permanent --zone=public --add-service=nfs

$ firewall-cmd --reload

# Display a list of the exported file systems.
$ showmount -e

# Displays all of the current clients and all of the file systems that the
clients have mounted.
$ showmount -a
```

3. Mount the NFSv4 file system with `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM`.

   See Attach Network File System to Autonomous Database for more information.

DBA_CLOUD_FILE_SYSTEMS View

The `DBA_CLOUD_FILE_SYSTEMS` view lists information about the network file system attached to a directory location in the database.

Column	Data Type	Description
`FILE_SYSTEM_NAME`	`VARCHAR2(128)`	File system name
`FILE_SYSTEM_LOCATION`	`VARCHAR2(4000)`	File system location
`DIRECTORY_NAME`	`VARCHAR2(128)`	Attached directory name
`DIRECTORY_PATH`	`VARCHAR2(4000)`	Attached directory path
`NFS_VERSION`	`NUMBER`	The NFS version.
`DESCRIPTION`	`VARCHAR2(4000)`	The value provided for the description parameter when you run `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM`
`CREATION_TIME`	`TIMESTAMP(6) WITH TIME ZONE`	Creation timestamp
`UPDATE_TIME`	`TIMESTAMP(6) WITH TIME ZONE`	Update timestamp

Load Data from Directories in Autonomous Database

As an alternative to an object store location URI, you can specify a directory with `DBMS_CLOUD` procedures to load or unload data from files in a local directory, including directories created on attached network file systems.

The following procedures support specifying files in a directory with the `file_uri_list` parameter:

- `DBMS_CLOUD.COPY_COLLECTION`
- `DBMS_CLOUD.COPY_DATA`

- DBMS_CLOUD.CREATE_EXTERNAL_TABLE

The following procedures support specifying files in a directory with the partitioning_clause parameter:

- DBMS_CLOUD.CREATE_EXTERNAL_TABLE
- DBMS_CLOUD.CREATE_HYBRID_PART_TABLE

You can specify one directory and one or more file names or use a comma separated list of directories and file names. The format to specify a directory is:'*MY_DIR*:*filename.ext*'. By default the directory name *MY_DIR* is a database object and is case-insensitive. The file name is case sensitive.

When you use the file_uri_list parameter to specify a directory you do not need to include the credential_name parameter, but you need READ object privileges on the directory.

For example, with a call to DBMS_CLOUD.COPY_DATA, use the file_uri_list parameter to specify files in a directory:

```
BEGIN
  DBMS_CLOUD.COPY_DATA(
     table_name => 'HRDATA1',
     file_uri_list => 'HR_DIR:test.csv',
     format => JSON_OBJECT('type' value 'csv')  );
END;
/
```

This example copies the data from test.csv in the local directory HR_DIR to the table HRDATA1.

Regular expressions are not supported when specifying the file names in a directory. You can only use wildcards to specify file names in a directory. The character "*" can be used as the wildcard for multiple characters, and the character "?" can be used as the wildcard for a single character. For example:'*MY_DIR*:*" or '*MY_DIR*:test?'

To specify multiple directories, use a comma separated list of directories: For example:'*MY_DIR1*:*, *MY_DIR2*:test?'

Use double quotes to specify a case-sensitive directory name. For example:'"*my_dir1*":*, "*my_dir2*":Test?'

To include a quote character, use two quotes. For example:'*MY_DIR*:''filename.ext'. This specifies the filename starts with a quote (').

See Attach Network File System to Autonomous Database for information on attaching network file systems.

Notes for Using Directories with DBMS_CLOUD Procedures

Note the following when you use DBMS_CLOUD procedures and specify a directory with the file_uri_list parameter:

- Compression options for files such as GZIP are not supported for directory files. See the compression format option in DBMS_CLOUD Package Format Options for more information.
- Special characters such as colon (:), single quote('), and comma(,) are not supported in the directory name.

Copy Files Between Object Store and a Directory in Autonomous Database

Use the procedure `DBMS_CLOUD.PUT_OBJECT` to copy a file from a directory to Object Store. Use the procedure `DBMS_CLOUD.GET_OBJECT` to copy a file from Object Store to a directory.

For example, to copy a file from Object Store to the `stage` directory, run the following command:

```
BEGIN
   DBMS_CLOUD.GET_OBJECT(
   credential_name => 'DEF_CRED_NAME',
   object_uri => 'https://objectstorage.usphoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/cwallet.sso',
   directory_name => 'STAGE');
END;
/
```

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

To run `DBMS_CLOUD.GET_OBJECT` with a user other than ADMIN you need to grant write privileges on the directory to that user.

To run `DBMS_CLOUD.PUT_OBJECT` with a user other than ADMIN you need to grant read privileges on the directory to that user.

See GET_OBJECT Procedure and Function and PUT_OBJECT Procedure for more information.

Bulk Operations for Files in the Cloud

The PL/SQL package `DBMS_CLOUD` offers parallel execution support for bulk file upload, download, copy, and transfer activities, which streamlines the user experience and delivers optimal performance for bulk file operations.

The package `DBMS_CLOUD` supports loading and unloading files into the following cloud services:

- Oracle Cloud Infrastructure Object Storage
- Azure Blob Storage
- Amazon S3
- Amazon S3-Compatible, including Wasabi Hot Cloud Storage
- Google Cloud Storage

See DBMS_CLOUD Package File URI Formats for more information.

- About Bulk File Operations

- Bulk Copy Files in Cloud Object Storage
  Use the `DBMS_CLOUD.BULK_COPY` procedure to bulk copy files from one Object Store bucket or folder to another bucket or folder.
- Bulk Move Files Across Cloud Object Storage
- Bulk Download Files from Cloud Object Storage
  Use the `DBMS_CLOUD.BULK_DOWNLOAD` procedure to bulk download files from the Cloud Object Store location to the directory object in an Autonomous Database.
- Bulk Upload Files to Cloud Object Storage
  Use the `DBMS_CLOUD.BULK_UPLOAD` procedure to bulk upload files from a directory object in database to a Cloud Object Store location.
- Bulk Delete Files from Cloud Object Storage
  Use the `DBMS_CLOUD.BULK_DELETE` procedure to bulk delete files from Cloud Object Storage.
- Monitor and Troubleshoot Bulk File Loads
  All `DBMS_CLOUD` data load operations are logged in the `dba_load_operations` and `user_load_operations` views.
- Notes for Bulk File Operations
  Provides notes for the credentials you use with bulk file operations.

About Bulk File Operations

`DBMS_CLOUD` bulk file operations support downloading and uploading files on Autonomous Database, including support for copying, moving, and deleting files across Cloud Object Stores, either within the same Object Store provider or between Object Store providers. The bulk file operations support parallelism and provide optimal performance for uploading, downloading, copying, and moving files. Parallelism for bulk file operations is handled by specifying a priority for an operation. The supported priorities are:

- `HIGH`: Use the Autonomous Database instance CPU compute count to determine the number of parallel files processed.
- `MEDIUM`: Use the concurrency limit for Medium service to determine the parallelism.
- `LOW`: Process files in serial order (no concurrency).

Running bulk operations with a higher priority uses more database resources and operations complete faster when parallelism can speed up the operation. A lower priority consumes less database resources and operations take longer to complete when parallelism can speed up the operation. When bulk operations involve small numbers of files that contain little data, specifying higher priority may not change the performance.

To increase parallelism for bulk file operations use `HIGH` priority and increase the number of CPUs assigned to your Autonomous Database instance. The maximum supported concurrent file operations is limited to 64 for bulk file operations.

The default priority is `MEDIUM`, which specifies that bulk file operations use the concurrency limit defined for the MEDIUM consumer group. See Manage Concurrency and Priorities on Autonomous Database for more information.

See DBMS_CLOUD for Bulk File Management for details on using the `format` parameter to specify the `priority` with bulk file operations.

Bulk Copy Files in Cloud Object Storage

Use the `DBMS_CLOUD.BULK_COPY` procedure to bulk copy files from one Object Store bucket or folder to another bucket or folder.

1. Create a credential object to access the source location.

   The source credential name, as specified with the `source_credential_name` parameter is by default also used as the credential for the target location.

   See CREATE_CREDENTIAL Procedure for more information.

2. When the source and target are in distinct Object Stores or have different accounts with the same cloud provider, create a credential to access the target location and include the `target_credential_name` parameter.

3. Run `DBMS_CLOUD.BULK_COPY` procedure to copy files in parallel from one Object Store bucket or folder to another bucket or folder which can be across cloud provider, accounts, and buckets. To run the procedure, you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD`.

   ```
   BEGIN
   DBMS_CLOUD.BULK_COPY (
         source_credential_name => 'OCI_CRED',
         source_location_uri    => 'https://objectstorage.us-
   phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/o',
         target_location_uri    => 'https://objectstorage.us-
   phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/o',
         format                 => JSON_OBJECT ('logretention' value 7,
   'logprefix' value 'BULKOP')
   );
   END;
   /
   ```

   This example bulk copies files from one Oracle Cloud Infrastructure Object Storage bucket to another.

   See BULK_COPY Procedure for more information.

   See DBMS_CLOUD URI Formats for more information.

For example, use `DBMS_CLOUD.BULK_COPY` to copy files from Amazon S3 to Oracle Cloud Infrastructure Object Storage.

```
BEGIN
DBMS_CLOUD.BULK_COPY(
      source_credential_name  => 'AWS_CRED',
      source_location_uri     => 'https://bucketname.s3-us-
west-2.amazonaws.com/',
      target_credential_name  => 'OCI_CRED',
      target_location_uri     => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/o',
      format                  =>  JSON_OBJECT ('logretention' value 7,
'logprefix' value 'BULKOP')
);
END;
/
```

Bulk Move Files Across Cloud Object Storage

Use the `DBMS_CLOUD.BULK_MOVE` procedure to bulk move files from one Cloud Object Storage location to another.

The first step in moving files is copying them to the target location. After the files are successfully copied, they are deleted from the source location.

The files are renamed rather than copied if Object Store allows renaming operations between source and target locations.

1. Create a credential object to access the source location.

   The source credential name, as specified with the `source_credential_name` parameter is by default also used as the credential for the target location.

   See CREATE_CREDENTIAL Procedure for more information.

2. When the source and target are in distinct Object Stores or have different accounts with the same cloud provider, create a credential to access the target location and include the `target_credential_name` parameter.

3. Run `DBMS_CLOUD.BULK_MOVE` procedure to bulk move files from one Cloud Object Storage location to another. To run the procedure, you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD`.

   ```
   BEGIN
   DBMS_CLOUD.BULK_MOVE (
        source_credential_name => 'OCI_CRED',
        source_location_uri    => 'https://objectstorage.us-
   phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/o',
        target_location_uri    => 'https://objectstorage.us-
   phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/o',
        format                 => JSON_OBJECT ('logretention' value 7,
   'logprefix' value 'BULKMOVE')
   );
   END;
   /
   ```

   This example bulk moves files from one Oracle Cloud Infrastructure Object Storage location to another.

   See BULK_MOVE Procedure for more information.

   See DBMS_CLOUD URI Formats for more information.

For example, use `DBMS_CLOUD.BULK_MOVE` to move files from Amazon S3 to Oracle Cloud Infrastructure Object Storage.

```
BEGIN
DBMS_CLOUD.BULK_MOVE(
      source_credential_name  => 'AWS_CRED',
      source_location_uri     => 'https://bucketname.s3-us-
west-2.amazonaws.com/',
      target_credential_name  => 'OCI_CRED',
      target_location_uri     => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/o',
      format                  =>  JSON_OBJECT ('logretention' value 7,
```

```
'logprefix' value 'BULKOP')
);
END;
/
```

Bulk Download Files from Cloud Object Storage

Use the `DBMS_CLOUD.BULK_DOWNLOAD` procedure to bulk download files from the Cloud Object Store location to the directory object in an Autonomous Database.

1. Create a credential to access your Cloud Object Storage.

   The credential name, is specified with the `credential_name` parameter.

   See CREATE_CREDENTIAL Procedure for more information.

2. Run `DBMS_CLOUD.BULK_DOWNLOAD` procedure to download files into an Autonomous Database directory from Cloud Object Storage. To run the procedure, you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD`.

   ```
   BEGIN
   DBMS_CLOUD.BULK_DOWNLOAD (
        credential_name => 'OCI_CRED',
        location_uri    => 'https://objectstorage.us-
   phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
        directory_name  => 'BULK_TEST',
        format          => JSON_OBJECT ('logretention' value 7, 'logprefix'
   value 'BULKOP')
    );
   END;
   /
   ```

   This example bulk downloads files from the Oracle Cloud Infrastructure Object Store location URI to the directory object in an Autonomous Database.

   > **Note:**
   >
   > To write the files in the target directory object, you must have the `WRITE` privilege on the directory object.

   See BULK_DOWNLOAD Procedure for more information.

   See DBMS_CLOUD URI Formats for more information.

Bulk Upload Files to Cloud Object Storage

Use the `DBMS_CLOUD.BULK_UPLOAD` procedure to bulk upload files from a directory object in database to a Cloud Object Store location.

1. Create a credential to access your Cloud Object Storage.

   The credential name, is specified with the `credential_name` parameter.

   See CREATE_CREDENTIAL Procedure for more information.

2. Run `DBMS_CLOUD.BULK_UPLOAD` procedure to copy files into Cloud Object Storage from a database directory on your Autonomous Database instance. To run the procedure, you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD`.

```
BEGIN
DBMS_CLOUD.BULK_UPLOAD (
     credential_name => 'OCI_CRED',
     location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
     directory_name  => 'BULK_TEST',
     format          => JSON_OBJECT ('logretention' value 5, 'logprefix'
value 'BULKUPLOAD')
 );
END;
/
```

This example bulk uploads files from a directory object, as specified with the `directory_name` parameter to the Oracle Cloud Infrastructure Object Store location URI.

> **Note:**
>
> To read the source files in the directory object, you must have the `READ` privilege on the directory object.

See BULK_UPLOAD Procedure for more information.

Bulk Delete Files from Cloud Object Storage

Use the `DBMS_CLOUD.BULK_DELETE` procedure to bulk delete files from Cloud Object Storage.

1. Create a credential to access your Cloud Object Storage.

   The credential name, is specified with the `credential_name` parameter.

   The `credential_name` can be `NULL` for public or Pre-authenticated or Pre-signed bucket URI.

   See CREATE_CREDENTIAL Procedure for more information.

2. Run `DBMS_CLOUD.BULK_DELETE` procedure to delete files from the Cloud Object Store. To run the procedure, you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD`.

```
BEGIN
DBMS_CLOUD.BULK_DELETE (
     credential_name => 'OCI_CRED',
     location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
     format          => JSON_OBJECT ('logretention' value 5, 'logprefix'
value 'BULKDEL')
);
END;
/
```

This example bulk deletes files from the Oracle Cloud Infrastructure Object Store.

See BULK_DELETE Procedure for more information.

See DBMS_CLOUD URI Formats for more information.

Monitor and Troubleshoot Bulk File Loads

All `DBMS_CLOUD` data load operations are logged in the `dba_load_operations` and `user_load_operations` views.

You can use the following views to monitor and troubleshoot bulk file loads:

- `dba_load_operations`: shows all load operations.
- `user_load_operations`: shows the load operations in your schema.

Query these views to see information about ongoing and completed bulk file operations. For example, the following `SELECT` statement with a `WHERE` clause predicate on the `TYPE` shows `DOWNLOAD` operations:

```
SELECT owner_name, type, status, start_time, update_time, status_table, rows_loaded
   FROM user_load_operations WHERE type = 'DOWNLOAD';

OWNER_NAME TYPE     STATUS    START_TIME               UPDATE_TIME              STATUS_TABLE
ROWS_LOADED
---------- -------- --------- ------------------------ ------------------------ ------------
-----------
"ADMIN"    DOWNLOAD COMPLETED 2022-10-17T20:42:19.498Z 2022-10-17T20:42:21.054Z
DWN$2_STATUS           4
"ADMIN"    DOWNLOAD FAILED    2022-10-17T20:40:54.348Z 2022-10-17T20:40:55.679Z
DWN$1_STATUS
```

The `STATUS_TABLE` column shows the name of the table you can query to look at detailed logging information for the bulk download operation. For example:

```
DESCRIBE DWN$2_STATUS
Name          Null?    Type
------------- -------- ---------------------------
ID            NOT NULL NUMBER
NAME                   VARCHAR2(4000)
BYTES                  NUMBER
CHECKSUM               VARCHAR2(128)
LAST_MODIFIED          TIMESTAMP(6) WITH TIME ZONE
STATUS                 VARCHAR2(30)
ERROR_CODE             NUMBER
ERROR_MESSAGE          VARCHAR2(4000)
START_TIME             TIMESTAMP(6) WITH TIME ZONE
END_TIME               TIMESTAMP(6) WITH TIME ZONE
SID                    NUMBER
SERIAL#                NUMBER
ROWS_LOADED            NUMBER

SELECT id, name, bytes, status, error_message, start_time, end_time FROM
DWN$2_STATUS;
ID NAME       BYTES STATUS    ERROR_MESSAGE START_TIME
```

```
END_TIME
-- ---------- ----- --------- ------------- -----------------------
-----------------------
 1 trees.txt     58 COMPLETED               2022-10-17T20:42:19.998Z
2022-10-17T20:42:20.421Z
 2 trees1.txt    58 COMPLETED               2022-10-17T20:42:20.425Z
2022-10-17T20:42:20.533Z
 3 trees2.txt    58 COMPLETED               2022-10-17T20:42:20.535Z
2022-10-17T20:42:20.894Z
 4 trees3.txt    58 COMPLETED               2022-10-17T20:42:20.896Z
2022-10-17T20:42:21.043Z
```

The status table shows each file name and its status for the bulk operation.

The relevant error number and message are recorded in the status table if an operation on a specific file fails.

For completed operations, the time needed for each operation can be calculated using the reported `START_TIME` and `END_TIME` time.

The file operation `STATUS` column can have one of the following values:

File Status	Description
`COMPLETED`	File operation completed successfully.
`FAILED`	File operation failed, a retry may be attempted for two times.
`PENDING`	The file operation has not yet started.
`RUNNING`	File operation is currently in progress.
`SKIPPED`	File operation skipped.

If any file operation fails after two retry attempts, then the bulk operation is marked as failed and an error is raised. For example:

```
ORA-20003: Operation failed, please query table DOWNLOAD$2_STATUS for error details
```

When you use a `DBMS_CLOUD` bulk file operation there are `format` parameter options that control status tables:

- `logretention`: Specifies an integer value that determines the duration in days that the status table is retained. The default value is 2 days.
- `logprefix`: Specifies a string value that determines the bulk operation status table's name prefix.

  Each bulk operation has its own default value for the `logprefix` option:

Procedure	Default Value for `logprefix`
`DBMS_CLOUD.BULK_COPY`	`COPYOBJ`
`DBMS_CLOUD.BULK_DELETE`	`DELETE`
`DBMS_CLOUD.BULK_DOWNLOAD`	`DOWNLOAD`
`DBMS_CLOUD.BULK_MOVE`	`MOVE`
`DBMS_CLOUD.BULK_UPLOAD`	`UPLOAD`

See DELETE_ALL_OPERATIONS Procedure for information on clearing the `user_load_operations` table.

Notes for Bulk File Operations

Provides notes for the credentials you use with bulk file operations.

- You may specify principals as the `credential_name`, `source_credential_name` or `target_credential_name` parameter to access cloud resources securely without storing user credentials. The supported principals are:
  - Oracle Cloud Infrastructure Resource Principals
  - AWS Amazon Resource Names (ARN)s
  - Azure service principal
  - Google Service Account

  See Configure Policies and Roles to Access Resources for more information.
- The `credential_name`, `source_credential_name` or the `target_credential_name` can be `NULL` for public, Pre-authenticated, or Pre-signed bucket URI.

Replicate Data

Describes using Oracle GoldenGate to replicate data between an Autonomous Database instance and to any target database or platform that Oracle GoldenGate supports.

- Use Oracle GoldenGate to Replicate Data to Autonomous Database
  You can replicate data to Autonomous Database using Oracle GoldenGate.

Use Oracle GoldenGate to Replicate Data to Autonomous Database

You can replicate data to Autonomous Database using Oracle GoldenGate.

Oracle GoldenGate Capture for Oracle Autonomous Database supports the following:

- Replication for different use cases: Report Offloading, Active-Active, Cloud to Cloud, and Cloud to on-premise.
- Inter-region and cross-region replication: Replicate data between different Oracle Cloud data centers around the world.
- Replicate between targets: Replicate from an Autonomous Database to any target database or platform that Oracle GoldenGate supports, including to other Oracle Autonomous Database environments.

The following are not available for Always Free Autonomous Databases:

- Supplemental logging
- Oracle GoldenGate Extract

See Supplemental Logging for more information.

See Configure Replicat to Apply to Autonomous Databases for more information.

Exporting Data from Autonomous Database to Object Store or to Other Oracle Databases

To export data from an Autonomous Database, use one of the following methods:

- Export to Cloud Object Store:
  - Use the procedure `DBMS_CLOUD.EXPORT_DATA` to export data to your Cloud Object Store as text, by specifying a query. This export method supports exporting data as CSV, JSON, Parquet, or XML files.

    This export method supports all the Cloud Object Stores supported by Autonomous Database.
  - Use Oracle Data Pump to export data to a directory on your database, and then move the data from the directory to Cloud Object Storage.
  - Use Oracle Data Pump to export the data to Cloud Object Storage directly. This method is only supported with Oracle Cloud Infrastructure Object Storage and Oracle Cloud Infrastructure Object Storage Classic.
  - Use the procedure `DBMS_CLOUD.EXPORT_DATA` to export data to your Cloud Object Store as Oracle Data Pump dump files, by specifying a query. This method is only supported with Oracle Cloud Infrastructure Object Storage and Oracle Cloud Infrastructure Object Storage Classic.
- Export to a directory:
  - Use the procedure `DBMS_CLOUD.EXPORT_DATA` to export data to a directory as text by specifying a query. This export method supports exporting data as CSV, JSON, Parquet, or XML files.
  - Use Oracle Data Pump to export data to a directory.
- Use Oracle GoldenGate Capture for Oracle Autonomous Database.

Topics

- Download Dump Files, Run Data Pump Import, and Clean Up Object Store
  If required, download the dump files from Cloud Object Store and use Oracle Data Pump Import to import the dump file set to the target database. Then perform any required clean up.
- Access Log Files for Data Pump Export
  The log files for Data Pump Export operations are stored in the directory you specify with the data pump `directory` parameter.
- Oracle GoldenGate Capture for Oracle Autonomous Database
  Using Oracle GoldenGate you can capture changes from an Oracle Autonomous Database and replicate to any target database or platform that Oracle GoldenGate supports, including another Oracle Autonomous Database.

Export Data to Object Store as Text

Use `DBMS_CLOUD.EXPORT_DATA` to export data as text from an Autonomous Database to cloud Object Store. The text format export options are CSV, JSON, Parquet, or XML.

See Move Data to Object Store as Oracle Data Pump Files Using EXPORT_DATA for details on exporting to cloud Object Store in Oracle Data Pump dump format.

- Export JSON Data to Cloud Object Storage
  Shows the steps to export table data from your Autonomous Database to Cloud Object Storage as JSON data by specifying a query.
- Export Data as CSV to Cloud Object Storage
  Shows the steps to export table data from your Autonomous Database to Cloud Object Storage as CSV data by specifying a query.
- Export Data as Parquet to Cloud Object Storage
  Shows the steps to export table data from your Autonomous Database to Cloud Object Storage as Parquet data by specifying a query.
- Export Data as XML to Cloud Object Storage
  Shows the steps to export table data from your Autonomous Database to Cloud Object Storage as XML data by specifying a query.
- File Naming for Text Output (CSV, JSON, Parquet, or XML)
  Describes the output file naming using `DBMS_CLOUD.EXPORT_DATA` with CSV, JSON, Parquet, or XML text file output.

Export JSON Data to Cloud Object Storage

Shows the steps to export table data from your Autonomous Database to Cloud Object Storage as JSON data by specifying a query.

This export method supports all the Cloud Object Stores supported by Autonomous Database, and you can use an Oracle Cloud Infrastructure resource principal to access your Oracle Cloud Infrastructure Object Store, Amazon Resource Names (ARNs) to access AWS Simple Storage Service (S3), an Azure service principal to access Azure BLOB storage, or a Google service account to access Google Cloud Platform (GCP) resources.

1. Connect to your Autonomous Database instance.

   See Connect to Autonomous Database for more information.
2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'user1@example.com',
    password => 'password'
  );
END;
/
```

The values you provide for `username` and `password` depend on the Cloud Object Storage service you are using.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the `format` parameter `type` with the value `json` to export the results as JSON files on Cloud Object Storage.

   To generate the JSON output files there are two options for the `file_uri_list` parameter:

   - Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage.
   - Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage and include a file name prefix to use when generating the file names for the exported JSON.

   If you do not include the file name prefix in the `file_uri_list`, `DBMS_CLOUD.EXPORT_DATA` supplies a file name prefix. See File Naming for Text Output (CSV, JSON, Parquet, or XML) for details.

   For example, the following shows `DBMS_CLOUD.EXPORT_DATA` with a file name prefix specified in `file_uri_list`:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/dept_export',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'json'));
END;
/
```

   In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

   When record delimiters include escape characters, such as \r\n or \t, enclose the record delimiters in double quotes. For example, to use the record delimiter `\r\n`, enclose the value in double quotes:`"\r\n"`.

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
```

```
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/dept_export',
    query => 'SELECT * FROM DEPT',
    format => JSON_OBJECT('type' value 'json', 'recorddelimiter' value
'"\r\n"' format json));
END;
/
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.

For detailed information about the parameters, see EXPORT_DATA Procedure.

For detailed information about the available `format` parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see DBMS_CLOUD Package Format Options for EXPORT_DATA.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The `query` parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the `format` parameter with the `compression` option to compress the output files.
- Specify the `format` parameter with the `encryption` option to encrypt data while exporting. See Encrypt Data While Exporting to Object Storage for more information.
- When you no longer need the files that you export, use the procedure `DBMS_CLOUD.DELETE_OBJECT` or use native Cloud Object Storage commands to delete the files.

Export Data as CSV to Cloud Object Storage

Shows the steps to export table data from your Autonomous Database to Cloud Object Storage as CSV data by specifying a query.

This export method supports all the Cloud Object Stores supported by Autonomous Database, and you can use an Oracle Cloud Infrastructure resource principal to access your Oracle Cloud Infrastructure Object Store, Amazon Resource Names (ARNs) to access AWS Simple Storage Service (S3), an Azure service principal to access Azure BLOB storage, or a Google service account to access Google Cloud Platform (GCP) resources.

1. Connect to your Autonomous Database instance.

   See Connect to Autonomous Database for more information.

2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'user1@example.com',
    password => 'password'
  );
END;
/
```

The values you provide for `username` and `password` depend on the Cloud Object Storage service you are using.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the `format` parameter `type` with the value `csv` to export the results as CSV files on Cloud Object Storage.

   To generate the CSV output files there are two options for the `file_uri_list` parameter:

   - Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage.
   - Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage and include a file name prefix to use when generating the file names for the exported CSV files.

   If you do not include the file name prefix in the `file_uri_list`, `DBMS_CLOUD.EXPORT_DATA` supplies a file name prefix. See File Naming for Text Output (CSV, JSON, Parquet, or XML) for details.

   For example, the following shows `DBMS_CLOUD.EXPORT_DATA` with a file name prefix specified in `file_uri_list`:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/dept_export',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'csv', 'delimiter' value
'|', 'compression' value 'gzip'));
END;
/
```

   In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

   For detailed information about the parameters, see EXPORT_DATA Procedure.

   For detailed information about the available `format` parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see DBMS_CLOUD Package Format Options for EXPORT_DATA.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The `query` parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the `format` parameter with the `compression` option to compress the output files.
- Specify the `format` parameter with the `encryption` option to encrypt data while exporting. See Encrypt Data While Exporting to Object Storage for more information.
- When you no longer need the files that you export, use the procedure `DBMS_CLOUD.DELETE_OBJECT` or use native Cloud Object Storage commands to delete the files.

Export Data as Parquet to Cloud Object Storage

Shows the steps to export table data from your Autonomous Database to Cloud Object Storage as Parquet data by specifying a query.

This export method supports all the Cloud Object Stores supported by Autonomous Database, and you can use an Oracle Cloud Infrastructure resource principal to access your Oracle Cloud Infrastructure Object Store, Amazon Resource Names (ARNs) to access AWS Simple Storage Service (S3), an Azure service principal to access Azure BLOB storage, or a Google service account to access Google Cloud Platform (GCP) resources.

1. Connect to your Autonomous Database instance.

   See Connect to Autonomous Database for more information.

2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DEF_CRED_NAME',
       username => 'user1@example.com',
       password => 'password'
     );
   END;
   /
   ```

   The values you provide for `username` and `password` depend on the Cloud Object Storage service you are using.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the `format` parameter `type` with the value `parquet` to export the results as parquet files on Cloud Object Storage.

   To generate the parquet output files there are two options for the `file_uri_list` parameter:

   - Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage.
   - Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage and include a file name prefix to use when generating the file names for the exported parquet files.

   If you do not include the file name prefix in the `file_uri_list`, `DBMS_CLOUD.EXPORT_DATA` supplies a file name prefix. See File Naming for Text Output (CSV, JSON, Parquet, or XML) for details.

   For example, the following shows `DBMS_CLOUD.EXPORT_DATA` with a file name prefix specified in `file_uri_list`:

   ```
   BEGIN
     DBMS_CLOUD.EXPORT_DATA(
       credential_name => 'DEF_CRED_NAME',
       file_uri_list   => 'https://objectstorage.us-
   ```

```
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/dept_export',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'parquet', 'compression'
value 'snappy'));
END;
/
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.

For detailed information about the parameters, see EXPORT_DATA Procedure.

For detailed information about the available `format` parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see DBMS_CLOUD Package Format Options for EXPORT_DATA.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The `query` parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the `format` parameter with the `compression` option to compress the output files. The default `compression` for `type parquet` is `snappy`.
- When you no longer need the files that you export, use the procedure `DBMS_CLOUD.DELETE_OBJECT` or use native Cloud Object Storage commands to delete the files.
- See DBMS_CLOUD Package Oracle Data Type to Parquet Mapping for details on Oracle Type to Parquet Type mapping.

  The following types are not supported or have limitations on their support for exporting Parquet with `DBMS_CLOUD.EXPORT_DATA`:

Oracle Type	Notes
`BFILE`	Not supported
`BLOB`	Not supported
`DATE`	Supported with the following limitation: `DATE` format supports only date, month and year. Hour, minute and seconds are not supported. See DBMS_CLOUD Package Oracle Data Type to Parquet Mapping for details on NLS format limitations for exporting `DATE` to Parquet.
`INTERVAL DAY TO SECOND`	Supported and is treated as string internally
`INTERVAL YEAR TO MONTH`	Supported and is treated as string internally
`LONG`	Not supported
`LONG RAW`	Not supported

Oracle Type	Notes
NUMBER	Supported with the following limitations: – Can have maximum precision of 38 and scale equal to less than precision. – If no precision and scale is provided for the column NUMBER type, by default precision of 38 and scale of 20 is used. – Negative scale is not supported for NUMBER types.
Object Types	Not supported
TIMESTAMP	Supported with the following limitations: – If there are multiple columns with different precision, highest precision will be taken. – TIMESTAMP WITH TIME ZONE Oracle datatype will use the timestamp only. See DBMS_CLOUD Package Oracle Data Type to Parquet Mapping for details on NLS format limitations for exporting TIMESTAMP to Parquet.

Export Data as XML to Cloud Object Storage

Shows the steps to export table data from your Autonomous Database to Cloud Object Storage as XML data by specifying a query.

This export method supports all the Cloud Object Stores supported by Autonomous Database, and you can use an Oracle Cloud Infrastructure resource principal to access your Oracle Cloud Infrastructure Object Store, Amazon Resource Names (ARNs) to access AWS Simple Storage Service (S3), an Azure service principal to access Azure BLOB storage, or a Google service account to access Google Cloud Platform (GCP) resources.

1. Connect to your Autonomous Database instance.

   See Connect to Autonomous Database for more information.

2. Store your Cloud Object Storage credential using DBMS_CLOUD.CREATE_CREDENTIAL.

   For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'user1@example.com',
    password => 'password'
  );
END;
/
```

   The values you provide for username and password depend on the Cloud Object Storage service you are using.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Run DBMS_CLOUD.EXPORT_DATA and specify the format parameter type with the value xml to export the results as XML files on Cloud Object Storage.

To generate the XML output files there are two options for the `file_uri_list` parameter:

- Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage.
- Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage and include a file name prefix to use when generating the file names for the exported JSON.

If you do not include the file name prefix in the `file_uri_list`, `DBMS_CLOUD.EXPORT_DATA` supplies a file name prefix. See File Naming for Text Output (CSV, JSON, Parquet, or XML) for details.

For example, the following shows `DBMS_CLOUD.EXPORT_DATA` with a file name prefix specified in `file_uri_list`:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/dept_export',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'xml', 'compression' value
'gzip'));
END;
/
```

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

For detailed information about the parameters, see EXPORT_DATA Procedure.

For detailed information about the available `format` parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see DBMS_CLOUD Package Format Options for EXPORT_DATA.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The `query` parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the `format` parameter with the `compression` option to compress the output files.
- Specify the `format` parameter with the `encryption` option to encrypt data while exporting. See Encrypt Data While Exporting to Object Storage for more information.
- When you no longer need the files that you export, use the procedure `DBMS_CLOUD.DELETE_OBJECT` or use native Cloud Object Storage commands to delete the files.

File Naming for Text Output (CSV, JSON, Parquet, or XML)

Describes the output file naming using `DBMS_CLOUD.EXPORT_DATA` with CSV, JSON, Parquet, or XML text file output.

`DBMS_CLOUD.EXPORT_DATA` performs the query specified with the `query` parameter and sends the results to text files either in the Cloud Object Store bucket or to a directory. The output format depends on the `format` parameter `type` you specify (one of CSV, JSON, Parquet, or XML).

To speed up the procedure and to generate the output as fast as possible, `DBMS_CLOUD.EXPORT_DATA` divides its work. This means that, depending on system resources, when you run `DBMS_CLOUD.EXPORT_DATA` the procedure creates multiple output files either in the Cloud Object Store bucket or in the directory.

The format for each generated file is:

[*FileNamePrefix* | *client_info_module_action*]*_sequenceNum_timestamp.format_extension.*[*compression_extension*]

- *FileNamePrefix*: (optional) If a *FileNamePrefix* is supplied, `DBMS_CLOUD.EXPORT_DATA` uses the file name prefix to generate file names for the results. The *FileNamePrefix* is specified using the text supplied after the bucket or directory name in the `file_uri_list` parameter value.

  You cannot provide multiple values for the *FileNamePrefix* in the `file_uri_list`.

- *client_info_module_action*: If a file name prefix is not supplied with the `file_uri_list` parameter, `DBMS_CLOUD.EXPORT_DATA` uses the combination of *client_info*, application *module* and *action* as the file name prefix (when this information is available). The procedure obtains these names from the application information for the database session that runs the query. See DBMS_APPLICATION_INFO for information on *client_info*, *module* name, and *action* name.

  If a file name prefix is not supplied with the `file_uri_list` and the database session attributes are not available, `DBMS_CLOUD.EXPORT_DATA` uses the file name prefix "`data`".

- *sequenceNum*: The sequence number associated with the `DBMS_CLOUD.EXPORT_DATA` query. Depending on the query, the database service, and the number of ECPUs (OCPUs if your database uses OCPUs) there are one or more *sequenceNum*s. Also, depending on the size of the results, there are one or more output files for each *sequenceNum*.

  See Manage Concurrency and Priorities on Autonomous Database for information on database services.

- *timestamp*: Timestamp when the file is uploaded.

- *format_extension*: The default value depends on the `format type` value:

  - CSV format: `.csv`
  - JSON format: `.json`
  - PARQUET format `.parquet`
  - XML format: `.xml`

  For more information, see the description for `format` option `fileextension` in DBMS_CLOUD Package Format Options for EXPORT_DATA.

- *compression_extension*: When you include the `format` parameter with the `compression` option with the value `gzip`, this is "`gz`".

  When the `format type` is `parquet`, the `compression` value `snappy` is also supported and is the default.

For example, the file name prefix in the following `DBMS_CLOUD.EXPORT_DATA` procedure is specified in the `file_uri_list` parameter, as `dept_export`. The example generates the output to the provided Object Store in the specified format.

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
```

```
    file_uri_list   => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/dept_export',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'json'));
END;
/
```

When you specify a file name prefix the generated output files include the file name prefix, similar to the following:

```
dept_export_1_20210809T173033Z.json
dept_export_2_20210809T173034Z.json
dept_export_3_20210809T173041Z.json
dept_export_4_20210809T173035Z.json
```

The number of generated output files depends on the size of the results, the database service, and the number of ECPUs (OCPUs if your database uses OCPUs) in the Autonomous Database instance.

In the following example the `file_uri_list` parameter does not include a file name prefix and the `compression` parameter is supplied, with value `gzip`:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/',
    query           => 'SELECT * FROM DEPT',
    format          => json_object('type' value 'json', 'compression' value
'gzip'));
END;
/
```

When a file name prefix is not in the `file_uri_list` parameter, `DBMS_CLOUD.EXPORT_DATA` uses a file name prefix of the form: *client_info_module_action*. For this example the generated output files include the file name prefix that `DBMS_CLOUD.EXPORT_DATA` supplies and the files are compressed with gzip and the file extension `.gz` is added, as follows:

```
Client1_Module1_Action1_1_20210809T173033Z.json.gz
Client1_Module1_Action1_2_20210809T173034Z.json.gz
Client1_Module1_Action1_3_20210809T173041Z.json.gz
Client1_Module1_Action1_4_20210809T173035Z.json.gz
```

If the *client_info_module_action* session information is not available when you run `DBMS_CLOUD.EXPORT_DATA`, the file name prefix is set to `data`. For example:

```
data_1_20210809T173033Z.json.gz
data_2_20210809T173034Z.json.gz
data_3_20210809T173041Z.json.gz
data_4_20210809T173035Z.json.gz
```

For example, the file name prefix in the following `DBMS_CLOUD.EXPORT_DATA` procedure is specified in the `file_uri_list` parameter, as `dept_export`. The example generates the output to the provided directory in the specified format.

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    file_uri_list   => 'DATA_PUMP_DIR:sales.json',
    query           => 'SELECT * FROM SALES',
    format          => JSON_OBJECT('type' value 'json'));
END;
/
```

When you specify a file name prefix the generated output file include the file name prefix, similar to the following:

```
sales_1_20230705T124523275915Z.csv
```

Notes for file naming with `DBMS_CLOUD.EXPORT_DATA`:

- `DBMS_CLOUD.EXPORT_DATA` does not create buckets or directories.
- The number of files that `DBMS_CLOUD.EXPORT_DATA` generates is determined by the number of ECPUs (OCPUs if your database uses OCPUs), the database service, and the size of the result data.
- The following applies when providing a directory object name in the `file_uri_list` parameter:
  - The provided directory must exist and you must have `WRITE` access to the directory.
  - The directory name is case-sensitive when enclosed in double quotes.
  - The credential name parameter must not be provided.
- For CSV, JSON, or XML output, by default when a generated file contains 10MB of data a new output file is created. However, if you have less than 10MB of result data you may have multiple output files, depending on the database service and the number of ECPUs (OCPUs if your database uses OCPUs) for the Autonomous Database instance.

  The default output file chunk size is 10MB for CSV, JSON, or XML. You can change this value with the `format` parameter `maxfilesize` option. See DBMS_CLOUD Package Format Options for EXPORT_DATA for more information.
- For Parquet output, each generated file is less than 128MB and multiple output files may be generated. However, if you have less than 128MB of result data, you may have multiple output files depending on the database service and the number of ECPUs (OCPUs if your database uses OCPUs) for the Autonomous Database instance.

  The `format` parameter `maxfilesize` option does not apply for Parquet files.

Export Data to a Directory

Use `DBMS_CLOUD.EXPORT_DATA` to export files to a directory.

The directory you export files to can be in the Autonomous Database file system or an attached Network File System. See Creating and Managing Directories on Autonomous Database for more information.

- Export Data as CSV to a Directory
  Shows the steps to export table data from your Autonomous Database to a directory as CSV data by specifying a query.
- Export Data as JSON a Directory
  Shows the steps to export table data from your Autonomous Database to a directory as JSON data by specifying a query.
- Export Data as Parquet to a Directory
  Shows the steps to export table data from your Autonomous Database to a directory as Parquet data by specifying a query.
- Export Data as XML to a Directory
  Shows the steps to export table data from your Autonomous Database to Directory as XML data by specifying a query.
- Export Data to a Directory as Oracle Data Pump Files
  You can export data to a directory as Oracle Data Pump dump files by specifying a query.

Export Data as CSV to a Directory

Shows the steps to export table data from your Autonomous Database to a directory as CSV data by specifying a query.

1. Connect to your Autonomous Database instance.

   See Connect to Autonomous Database for more information.

2. Create a directory.

   For example:

   ```
   CREATE DIRECTORY export_dir AS 'export_dir';
   ```

   See Create Directory in Autonomous Database for more information.

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the `format` parameter `type` with the value `json` to export the results as CSV files to a directory. Do not include the `credential` parameter when sending output to a directory.

   ```
   BEGIN
    DBMS_CLOUD.EXPORT_DATA(
       file_uri_list => 'export_dir:sales.csv',
       format => JSON_OBJECT('type' value 'csv'),
       query => 'SELECT * FROM sales'
    );
   END;
   /
   ```

   When record delimiters include escape characters, such as \r\n or \t, enclose the record delimiters in double quotes. For example, to use the record delimiter `\r\n`, enclose the value in double quotes:`"\r\n"`.

   ```
   BEGIN
     DBMS_CLOUD.EXPORT_DATA(
       file_uri_list => 'export_dir:sales.csv',
       query => 'SELECT * FROM sales',
       format => JSON_OBJECT('type' value 'json', 'recorddelimiter' value
   ```

```
'"\r\n"' format json));
END;
/
```

The directory name is case-sensitive when the directory name is enclosed in double quotes. For example:

```
BEGIN
 DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => '"export_dir":sales.csv',
    format => JSON_OBJECT('type' value 'csv'),
    query => 'SELECT * FROM sales'
 );
END;
/
```

For detailed information about the parameters, see EXPORT_DATA Procedure.

For detailed information about the available `format` parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see DBMS_CLOUD Package Format Options for EXPORT_DATA.

The parameters are:

- `file_uri_list`: is a comma delimited list of the export file(s). Use of wildcard and substitution characters is not supported in the `file_uri_list`.
- `format`: specifies the required `type` parameter. The valid values are `datapump`, `json`, `xml`, `csv` and `parquet` and it also optionally defines the options you can specify for the export with the `ORACLE_DATAPUMP` Access Driver.
- `query`: specifies a `SELECT` statement so that only the required data is exported. The query determines the contents of the dump file(s).

> **Note:**
>
> The `DBMS_CLOUD.EXPORT_DATA` procedure creates the dump file(s) that you specify in the `file_uri_list`. The procedure does not overwrite files. If a dump file in the `file_uri_list` exists, `DBMS_CLOUD.EXPORT_DATA` generates another file with a unique name. `DBMS_CLOUD.EXPORT_DATA` does not create directories.

For detailed information about the parameters, see EXPORT_DATA Procedure.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The `query` parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the `format` parameter with the `compression` option to compress the output files.

Export Data as JSON a Directory

Shows the steps to export table data from your Autonomous Database to a directory as JSON data by specifying a query.

1. Connect to your Autonomous Database instance.

See Connect to Autonomous Database for more information.

2. Create a directory.

   For example:

   ```
   CREATE DIRECTORY export_dir AS 'export_dir';
   ```

   See Create Directory in Autonomous Database for more information.

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the `format` parameter `type` with the value `json` to export the results as JSON files to a directory. Do not include the `credential` parameter when sending output to a directory.

   ```
   BEGIN
    DBMS_CLOUD.EXPORT_DATA(
       file_uri_list => 'export_dir:sales.dmp',
       format => json_object('type' value 'json'),
       query => 'SELECT * FROM sales'
    );
   END;
   /
   ```

   When record delimiters include escape characters, such as \r\n or \t, enclose the record delimiters in double quotes. For example, to use the record delimiter `\r\n`, enclose the value in double quotes:`"\r\n"`.

   ```
   BEGIN
     DBMS_CLOUD.EXPORT_DATA(
       file_uri_list => 'export_dir:sales.dmp',
       query => 'SELECT * FROM sales',
       format => JSON_OBJECT('type' value 'json', 'recorddelimiter' value
   '"\r\n"' format json));
   END;
   /
   ```

   The directory name is case-sensitive when the directory name is enclosed in double quotes. For example:

   ```
   BEGIN
    DBMS_CLOUD.EXPORT_DATA(
       file_uri_list => '"export_dir":sales.dmp',
       format => json_object('type' value 'json'),
       query => 'SELECT * FROM sales'
    );
   END;
   /
   ```

   For detailed information about the parameters, see EXPORT_DATA Procedure.

   For detailed information about the available `format` parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see DBMS_CLOUD Package Format Options for EXPORT_DATA.

   The parameters are:

- `file_uri_list`: is a comma delimited list of the export file(s). Use of wildcard and substitution characters is not supported in the `file_uri_list`.
- `format`: specifies the required `type` parameter. The valid values are `datapump`, `json`, `xml`, `csv` and `parquet` and it also optionally defines the options you can specify for the export with the `ORACLE_DATAPUMP` Access Driver.
- `query`: specifies a `SELECT` statement so that only the required data is exported. The query determines the contents of the dump file(s).

> **Note:**
>
> The `DBMS_CLOUD.EXPORT_DATA` procedure creates the dump file(s) that you specify in the `file_uri_list`. The procedure does not overwrite files. If a dump file in the `file_uri_list` exists, `DBMS_CLOUD.EXPORT_DATA` generates another file with a unique name. `DBMS_CLOUD.EXPORT_DATA` does not create directories.

For detailed information about the parameters, see EXPORT_DATA Procedure.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The `query` parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the `format` parameter with the `compression` option to compress the output files.

Export Data as Parquet to a Directory

Shows the steps to export table data from your Autonomous Database to a directory as Parquet data by specifying a query.

1. Connect to your Autonomous Database instance.

   See Connect to Autonomous Database for more information.

2. Create a directory.

   For example:

   ```
   CREATE DIRECTORY export_dir AS 'export_dir';
   ```

   See Create Directory in Autonomous Database for more information.

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the `format` parameter `type` with the value `json` to export the results as Parquet files to a directory. Do not include the `credential` parameter when sending output to a directory.

   ```
   BEGIN
    DBMS_CLOUD.EXPORT_DATA(
       file_uri_list => 'export_dir:sales.parquet',
       format => JSON_OBJECT('type' value 'parquet'),
       query => 'SELECT * FROM sales'
    );
   END;
   /
   ```

The directory name is case-sensitive when the directory name is enclosed in double quotes. For example:

```
BEGIN
 DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => '"export_dir":sales.parquet',
    format => JSON_OBJECT('type' value 'parquet'),
    query => 'SELECT * FROM sales'
 );
END;
/
```

For detailed information about the parameters, see EXPORT_DATA Procedure.

For detailed information about the available `format` parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see DBMS_CLOUD Package Format Options for EXPORT_DATA.

The parameters are:

- `file_uri_list`: is a comma delimited list of the export file(s). Use of wildcard and substitution characters is not supported in the `file_uri_list`.
- `format`: specifies the required `type` parameter. The valid values are `datapump`, `json`, `xml`, `csv` and `parquet` and it also optionally defines the options you can specify for the export with the `ORACLE_DATAPUMP` Access Driver.
- `query`: specifies a `SELECT` statement so that only the required data is exported. The query determines the contents of the dump file(s).

For detailed information about the parameters, see EXPORT_DATA Procedure.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The `query` parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the `format` parameter with the `compression` option to compress the output files.
- See DBMS_CLOUD Package Oracle Data Type to Parquet Mapping for details on Oracle Type to Parquet Type mapping.

  The following types are not supported or have limitations on their support for exporting Parquet with `DBMS_CLOUD.EXPORT_DATA`:

Oracle Type	Notes
`BFILE`	Not supported
`BLOB`	Not supported
`DATE`	Supported with the following limitation: `DATE` format supports only date, month and year. Hour, minute and seconds are not supported. See DBMS_CLOUD Package Oracle Data Type to Parquet Mapping for details on NLS format limitations for exporting `DATE` to Parquet.
`INTERVAL DAY TO SECOND`	Supported and is treated as string internally
`INTERVAL YEAR TO MONTH`	Supported and is treated as string internally
`LONG`	Not supported

Oracle Type	Notes
LONG RAW	Not supported
NUMBER	Supported with the following limitations: – Can have maximum precision of 38 and scale equal to less than precision. – If no precision and scale is provided for the column NUMBER type, by default precision of 38 and scale of 20 is used. – Negative scale is not supported for NUMBER types.
Object Types	Not supported
TIMESTAMP	Supported with the following limitations: – If there are multiple columns with different precision, highest precision will be taken. – TIMESTAMP WITH TIME ZONE Oracle datatype will use the timestamp only. See DBMS_CLOUD Package Oracle Data Type to Parquet Mapping for details on NLS format limitations for exporting TIMESTAMP to Parquet.

Export Data as XML to a Directory

Shows the steps to export table data from your Autonomous Database to Directory as XML data by specifying a query.

1. Connect to your Autonomous Database instance.

   See Connect to Autonomous Database for more information.

2. Create a directory.

   For example:

   ```
   CREATE DIRECTORY export_dir AS 'export_dir';
   ```

   See Create Directory in Autonomous Database for more information.

3. Run DBMS_CLOUD.EXPORT_DATA and specify the format parameter type with the value json to export the results as XML files to a directory. Do not include the credential parameter when sending output to a directory.

   ```
   BEGIN
    DBMS_CLOUD.EXPORT_DATA(
       file_uri_list => 'export_dir:sales.csv',
       format => JSON_OBJECT('type' value 'csv'),
       query => 'SELECT * FROM sales'
    );
   END;
   /
   ```

   The directory name is case-sensitive when the directory name is enclosed in double quotes. For example:

   ```
   BEGIN
    DBMS_CLOUD.EXPORT_DATA(
   ```

```
    file_uri_list => '"export_dir":sales.xml',
    format => JSON_OBJECT('type' value 'xml'),
    query => 'SELECT * FROM sales'
 );
END;
/
```

For detailed information about the parameters, see EXPORT_DATA Procedure.

For detailed information about the available `format` parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see DBMS_CLOUD Package Format Options for EXPORT_DATA.

The parameters are:

- `file_uri_list`: is a comma delimited list of the export file(s). Use of wildcard and substitution characters is not supported in the `file_uri_list`.
- `format`: specifies the required `type` parameter. The valid values are `datapump`, `json`, `xml`, `csv` and `parquet` and it also optionally defines the options you can specify for the export with the `ORACLE_DATAPUMP` Access Driver.
- `query`: specifies a `SELECT` statement so that only the required data is exported. The query determines the contents of the dump file(s).

> **Note:**
>
> The `DBMS_CLOUD.EXPORT_DATA` procedure creates the dump file(s) that you specify in the `file_uri_list`. The procedure does not overwrite files. If a dump file in the `file_uri_list` exists, `DBMS_CLOUD.EXPORT_DATA` generates another file with a unique name. `DBMS_CLOUD.EXPORT_DATA` does not create directories.

For detailed information about the parameters, see EXPORT_DATA Procedure.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The `query` parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the `format` parameter with the `compression` option to compress the output files.

Export Data to a Directory as Oracle Data Pump Files

You can export data to a directory as Oracle Data Pump dump files by specifying a query.

With this export method you use the `DBMS_CLOUD.EXPORT_DATA` procedure to specify a query to select the data to export as a dump file to a directory.

1. Connect to your Autonomous Database instance.
2. Create a directory.

   For example:

   ```
   CREATE DIRECTORY export_dir AS 'export_dir';
   ```

   See Create Directory in Autonomous Database for more information.

3. Export data from Autonomous Database to your directory as Oracle Data Pump dump file(s) with `DBMS_CLOUD.EXPORT_DATA` and specify the `format` parameter `type` as `datapump`. For example:

```
BEGIN
 DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => 'export_dir:sales.dmp',
    format => json_object('type' value 'datapump'),
    query => 'SELECT * FROM sales'
 );
END;
/
```

Example to export data as multiple Data Pump files to a directory:

```
BEGIN
 DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => 'export_dir:sales1.dmp, export_dir:sales2.dmp',
    format => json_object('type' value 'datapump'),
    query => 'SELECT * FROM sales'
 );
END;
/
```

The parameters are:

- `file_uri_list`: is a comma delimited list of the export file(s). Use of wildcard and substitution characters is not supported in the `file_uri_list`.
- `format`: specifies the required `type` parameter. The valid values are `datapump`, `json`, `xml`, `csv` and `parquet` and it also optionally defines the options you can specify for the export with the `ORACLE_DATAPUMP` Access Driver.
- `query`: specifies a `SELECT` statement so that only the required data is exported. The query determines the contents of the dump file(s).

> **Note:**
>
> The `DBMS_CLOUD.EXPORT_DATA` procedure creates the dump file(s) that you specify in the `file_uri_list`.

For detailed information about the parameters, see EXPORT_DATA Procedure.

Notes for exporting data with `DBMS_CLOUD.EXPORT_DATA`:

- The provided directory must exist and you must be logged in as the `ADMIN` user or have `WRITE` access to the directory.
- The procedure does not overwrite files. If a dump file in the `file_uri_list` exists, `DBMS_CLOUD.EXPORT_DATA` reports an error such as:

```
ORA-31641: unable to create dump file  "/u02/exports/123.dmp"
ORA-27038: created file already exists
```

- `DBMS_CLOUD.EXPORT_DATA` does not create directories.
- The directory name is case-sensitive when enclosed in double quotes.
- The number of dump files that `DBMS_CLOUD.EXPORT_DATA` generates is determined when the procedure runs. The number of dump files that are generated depends on the number of file names you provide in the `file_uri_list` parameter, as well as on the number of ECPUs available to the instance, the service level, and the size of the data.

  For example, if you use a 2 ECPU Autonomous Database instance or the `low` service, then a single dump file is exported with no parallelism, even if you provide multiple file names. If you use an 8 ECPU Autonomous Database instance with the `medium` or `high` service, then the jobs can run in parallel and multiple dump files are exported if you provide multiple file names.
- The `query` parameter value that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- The dump files you create with `DBMS_CLOUD.EXPORT_DATA` cannot be imported using Oracle Data Pump `impdp`. Depending on the database, you can use these files as follows:
  - On an Autonomous Database, you can use the dump files with the `DBMS_CLOUD` procedures that support the `format` parameter `type` with the value '`datapump`'. You can import the dump files using `DBMS_CLOUD.COPY_DATA` or you can call `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` to create an external table.
  - On any other Oracle Database, such as Oracle Database 19c on-premise, you can import the dump files created with the procedure `DBMS_CLOUD.EXPORT_DATA` using the `ORACLE_DATAPUMP` access driver. See Unloading and Loading Data with the ORACLE_DATAPUMP Access Driver for more information.

Move Data with Data Pump Export to an Autonomous Database Directory

Oracle Data Pump offers very fast bulk data and metadata movement between Autonomous Database and other Oracle databases. To move data from an Autonomous Database to another Oracle database, use Oracle Data Pump to export to a directory on your Autonomous Database.

This export method is supported on all of the object stores that Autonomous Database supports:

- Oracle Cloud Infrastructure Object Storage
- Oracle Cloud Infrastructure Object Storage Classic
- Azure Blob Storage
- Amazon S3

> **Note:**
>
> If you are using Oracle Cloud Infrastructure Object Storage or Oracle Cloud Infrastructure Object Storage Classic, then you can use the alternative methods to export directly to object store. See Move Data with Data Pump Export to Object Store and Move Data to Object Store as Oracle Data Pump Files Using EXPORT_DATA for more information.

This export method includes the following steps:

1. Use Data Pump Export to save a dump file set to a directory on your database.

   See Use Data Pump to Create a Dump File Set on Autonomous Database for details.

2. Move the dump file set from the directory on your database to your cloud object store.

   See Move Dump File Set from Autonomous Database to Your Cloud Object Store for details.

3. Download the dump file set from the cloud object store, run Data Pump Import, and then perform any required clean up such as removing the dump file set from your cloud object store.

   See Download Dump Files, Run Data Pump Import, and Clean Up Object Store for details.

- Use Data Pump to Create a Dump File Set on Autonomous Database
  Shows the steps to export data from your Autonomous Database to a directory with Oracle Data Pump.
- Move Dump File Set from Autonomous Database to Your Cloud Object Store
  To move the dump file set to your Cloud Object Store, upload the files from the database directory to your Cloud Object Store.

Use Data Pump to Create a Dump File Set on Autonomous Database

Shows the steps to export data from your Autonomous Database to a directory with Oracle Data Pump.

Oracle recommends using the latest Oracle Data Pump version for exporting data from Autonomous Database to other Oracle databases, as it contains enhancements and fixes for a better experience. Download the latest version of Oracle Instant Client and download the Tools Package, which includes Oracle Data Pump, for your platform. See the installation instructions on the platform install download page for the installation steps required after you download Oracle Instant Client and the Tools Package. See Oracle Instant Client Downloads for details.

> **Note:**
>
> Database Actions provides a link for Oracle Instant Client. To access this link from Database Actions, under **Downloads**, click **Download Oracle Instant Client**.

1. Run Data Pump Export with the *dumpfile* parameter set, the *filesize* parameter set to less than 5G, and the *directory* parameter set. For example, the following shows how to export a schema named `SALES` in an Autonomous Database named `DB2022ADB` with 16 ECPUs:

```
expdp sales/password@db2022adb_high
directory=data_pump_dir
dumpfile=exp%U.dmp
parallel=4
encryption_pwd_prompt=yes
filesize=1G
logfile=export.log
```

> **Note:**
>
> If during the export with `expdp` you use the `encryption_pwd_prompt=yes` parameter then use `encryption_pwd_prompt=yes` with your import and input the same password at the `impdp` prompt to decrypt the dump files (remember the password you supply with export). The maximum length of the encryption password is 128 bytes.

For the best export performance use the `HIGH` database service for your export connection and set the `parallel` parameter to one quarter the number of ECPUs (.25 x **ECPU count**). If you are using OCPU compute model, set the parallel parameter to the number of OCPUs (1 x **OCPU count**). For information on which database service name to connect to run Data Pump Export, see Manage Concurrency and Priorities on Autonomous Database.

After the export is finished you can see the generated dump files by running the following query:

```
SELECT * FROM DBMS_CLOUD.LIST_FILES('DATA_PUMP_DIR');
```

For example, the output from this query shows the generated dump files and the export log file:

```
OBJECT_NAME                    BYTES  CHECKSUM                    CREATED
LAST_MODIFIED
------------------------ ---------- --------- -------------------------------
--------------------
exp01.dmp                      12288               12-NOV-19 06.10.47.0 PM GMT       12-
NOV-19...
exp02.dmp                       8192               12-NOV-19 06.10.48.0 PM GMT       12-
NOV-19...
exp03.dmp                    1171456               12-NOV-19 06.10.48.0 PM GMT       12-
NOV-19...
exp04.dmp                     348160               12-NOV-19 06.10.48.0 PM GMT       12-
NOV-19...
export.log                      1663               12-NOV-19 06.10.50.0 PM GMT       12-
NOV-19...
```

2. Move the dump file set to your cloud object store. See Move Dump File Set from Autonomous Database to Your Cloud Object Store for details.

Notes:

- To perform a full export or to export objects that are owned by other users, you need the `DATAPUMP_CLOUD_EXP` role.
- The `DATA_PUMP_DIR` is the only predefined directory. You can specify a different directory as the *directory* argument if you previously created the directory and you have write privileges on the directory. See Create Directory in Autonomous Database for information on creating directories.
- The API you use to move the dump files to Cloud Object Storage has a maximum file transfer size, so make sure you use a *filesize* argument that is less than or equal to the maximum supported size for your Cloud Object Storage service. See PUT_OBJECT Procedure for the Cloud Object Storage Service file transfer size limits.
- For more information on Oracle Data Pump Export see *Oracle Database Utilities*.

Move Dump File Set from Autonomous Database to Your Cloud Object Store

To move the dump file set to your Cloud Object Store, upload the files from the database directory to your Cloud Object Store.

1. Connect to your database.
2. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DEF_CRED_NAME',
       username => 'adb_user@example.com',
       password => 'password'
     );
   END;
   /
   ```

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name.

   See CREATE_CREDENTIAL Procedure for information about the `username` and `password` parameters for different object storage services.
3. Move the dump files from the database to your Cloud Object Store by calling `DBMS_CLOUD.PUT_OBJECT`.

   For example:

   ```
   BEGIN
      DBMS_CLOUD.PUT_OBJECT(credential_name => 'DEF_CRED_NAME',
        object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
   namespace-string/b/bucketname/o/exp01.dmp',
        directory_name => 'DATA_PUMP_DIR',
   ```

```
      file_name => 'exp01.dmp');
   DBMS_CLOUD.PUT_OBJECT(credential_name => 'DEF_CRED_NAME',
      object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/exp02.dmp',
      directory_name => 'DATA_PUMP_DIR',
      file_name => 'exp02.dmp');
   DBMS_CLOUD.PUT_OBJECT(credential_name => 'DEF_CRED_NAME',
      object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/exp03.dmp',
      directory_name => 'DATA_PUMP_DIR',
      file_name => 'exp03.dmp');
   DBMS_CLOUD.PUT_OBJECT(credential_name => 'DEF_CRED_NAME',
      object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/exp04.dmp',
      directory_name => 'DATA_PUMP_DIR',
      file_name => 'exp04.dmp');
END;
/
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.

See PUT_OBJECT Procedure for information on `PUT_OBJECT`.

4. Perform the required steps to use Oracle Data Pump import and clean up. See Download Dump Files, Run Data Pump Import, and Clean Up Object Store for more details.

Move Data with Data Pump Export to Object Store

To move data from Autonomous Databases to other Oracle databases you can use Oracle Data Pump. Using this export method you use Oracle Data Pump to directly export data to your object store. This export method is supported with Oracle Cloud Infrastructure Object Storage and Oracle Cloud Infrastructure Object Storage Classic.

See Move Data with Data Pump Export to an Autonomous Database Directory for details on the alternative export method to use with other supported cloud object stores.

To export data from Autonomous Databases to other Oracle databases, do the following:

1. Use Data Pump Export to export from your database to Object Storage.

   You have two options for setting the credential to access Object Store, depending on your Oracle Data Pump version:

   - Using Oracle Data Pump version 19.9 (or later) use the `CREDENTIAL` parameter to set the credential or set the database property `DEFAULT_CREDENTIAL`, both options are supported. See Use Oracle Data Pump to Export Data to Object Store Using CREDENTIAL Parameter (Version 19.9 or Later) for details.
   - For Oracle Data Pump versions before 19.9 the `CREDENTIAL` parameter is not supported. Use the `DEFAULT_CREDENTIAL` database property to set the credential to use to access your Object Store. See Use Oracle Data Pump to Export Data to Object Store Setting DEFAULT_CREDENTIAL Property for details.

2. Download the dump files from the Object Storage service, run Data Pump Import with the dump files, and perform any required clean up such as removing the dump file set from Object Storage.

See Download Dump Files, Run Data Pump Import, and Clean Up Object Store for details.

- Use Oracle Data Pump to Export Data to Object Store Using CREDENTIAL Parameter (Version 19.9 or Later)
  Shows the steps to export data from your database to Object Storage with Oracle Data Pump.
- Use Oracle Data Pump to Export Data to Object Store Setting DEFAULT_CREDENTIAL Property
  Shows the steps to export data from your database to Object Storage with Oracle Data Pump.

Use Oracle Data Pump to Export Data to Object Store Using CREDENTIAL Parameter (Version 19.9 or Later)

Shows the steps to export data from your database to Object Storage with Oracle Data Pump.

Oracle recommends using the latest Oracle Data Pump version for exporting data from Autonomous Database to other Oracle databases, as it contains enhancements and fixes for a better experience. Download the latest version of Oracle Instant Client and download the Tools Package, which includes Oracle Data Pump, for your platform from Oracle Instant Client Downloads. See the installation instructions on the platform install download page for the installation steps required after you download Oracle Instant Client and the Tools Package.

> **Note:**
>
> Database Actions provides a link for Oracle Instant Client. To access this link from Database Actions, under **Downloads**, click **Download Oracle Instant Client**.

If you are using Oracle Data Pump Version 19.9 or later, then you can use the `credential` parameter as shown in these steps. For instructions for using Oracle Data Pump Versions 19.8 and earlier, see Use Oracle Data Pump to Export Data to Object Store Setting DEFAULT_CREDENTIAL Property.

1. Connect to your database.
2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`. For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'user1@example.com',
    password => 'password'
  );
END;
/
```

The values you provide for `username` and `password` depend on the Cloud Object Storage service you are using.

If you are exporting to Oracle Cloud Infrastructure Object Storage, you can use the Oracle Cloud Infrastructure native URIs or Swift URIs, but the credentials must be auth tokens. See CREATE_CREDENTIAL Procedure for more information.

3. Run Data Pump Export with the `dumpfile` parameter set to the URL for an existing bucket on your Cloud Object Storage, ending with a file name or a file name with a substitution variable, such as `exp%U.dmp`, and with the `credential` parameter set to the name of the credential you created in the previous step. For example:

```
expdp admin/password@db2022adb_high \
     filesize=5GB \
     credential=def_cred_name \
     dumpfile=https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/exp%U.dmp \
     parallel=16 \
     encryption_pwd_prompt=yes \
     logfile=export.log \
     directory=data_pump_dir
```

> **Note:**
>
> If during the export with `expdp` you use the `encryption_pwd_prompt=yes` parameter then use `encryption_pwd_prompt=yes` with your import and input the same password at the `impdp` prompt to decrypt the dump files (remember the password you supply with export). The maximum length of the encryption password is 128 bytes.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

The `credential` parameter cannot be an OCI resource principal, Azure service principal, Amazon Resource Name (ARN), or a Google service account. See Accessing Cloud Resources by Configuring Policies and Roles for more information on resource principal based authentication.

For the best export performance use the `HIGH` database service for your export connection and set the `parallel` parameter to one quarter the number of ECPUs (.25 x **ECPU count**). If you are using OCPU compute model, set the parallel parameter to the number of OCPUs (1 x **OCPU count**).

For information on which database service name to connect to run Data Pump Export, see Database Service Names for Autonomous Data Warehouse.

For the dump file URL format for different Cloud Object Storage services, see DBMS_CLOUD URI Formats.

This example shows the recommended parameters for exporting from Autonomous Database. For these `expdp` parameters, note the following:

- The maximum `filesize` parameter value is 10000MB for Oracle Cloud Infrastructure Object Storage exports.
- The maximum `filesize` parameter value is 20GB for Oracle Cloud Infrastructure Object Storage Classic exports.
- If the specified `filesize` is too large, the export shows the error message:

  ```
  ORA-17500: ODM err:ODM HTTP Request Entity Too Large
  ```

- The `directory` parameter specifies the directory `data_pump_dir` for the specified log file, `export.log`. See Access Log Files for Data Pump Export for more information.

> **Note:**
>
> Oracle Data Pump divides each dump file part into smaller chunks for faster uploads. The Oracle Cloud Infrastructure Object Storage console shows multiple files for each dump file part that you export. The size of the actual dump files will be displayed as zero (0) and its related file chunks as 10mb or less. For example:
>
> ```
> exp01.dmp
> exp01.dmp_aaaaaa
> exp02.dmp
> exp02.dmp_aaaaaa
> ```
>
> Downloading the zero byte dump file from the Oracle Cloud Infrastructure console or using the Oracle Cloud Infrastructure CLI will not give you the full dump files. To download the full dump files from the Object Store, use a tool that supports Swift such as curl, and provide your user login and Swift auth token.
>
> ```
> curl -O -v -X GET -u 'user1@example.com:auth_token' \
>    https://swiftobjectstorage.us-ashburn-1.oraclecloud.com/v1/
> namespace-string/bucketname/exp01.dmp
> ```
>
> If you import a file with the `DBMS_CLOUD` procedures that support the `format` parameter `type` with the value '`datapump`', you only need to provide the primary file name. The procedures that support the '`datapump`' format type automatically discover and download the chunks.
>
> When you use `DBMS_CLOUD.DELETE_OBJECT`, the procedure automatically discovers and deletes the chunks when the procedure deletes the primary file.

4. Perform the required steps to use Oracle Data Pump import and clean up.

   See Download Dump Files, Run Data Pump Import, and Clean Up Object Store for details.

> **Note:**
>
> To perform a full export or to export objects that are owned by other users, you need the `DATAPUMP_CLOUD_EXP` role.

For detailed information on Oracle Data Pump Export parameters see *Oracle Database Utilities*.

Use Oracle Data Pump to Export Data to Object Store Setting DEFAULT_CREDENTIAL Property

Shows the steps to export data from your database to Object Storage with Oracle Data Pump.

Oracle recommends using the latest Oracle Data Pump version for exporting data from Autonomous Database to other Oracle databases, as it contains enhancements and fixes for a better experience. Download the latest version of Oracle Instant Client and download the Tools Package, which includes Oracle Data Pump, for your platform from Oracle Instant Client

Downloads. See the installation instructions on the platform install download page for the installation steps required after you download Oracle Instant Client and the Tools Package.

> **Note:**
>
> Database Actions provides a link for Oracle Instant Client. To access this link from Database Actions, under **Downloads**, click **Download Oracle Instant Client**.

1. Connect to your Autonomous Database.
2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`. For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'user1@example.com',
    password => 'password'
  );
END;
/
```

   The values you provide for `username` and `password` depend on the Cloud Object Storage service you are using.

   If you are exporting to Oracle Cloud Infrastructure Object Storage, you can use the Oracle Cloud Infrastructure native URIs or Swift URIs, but the credentials must be auth tokens. See CREATE_CREDENTIAL Procedure for more information.
3. As the ADMIN user, set the credential you defined in step 2 as the default credential for your database. For example:

```
ALTER DATABASE PROPERTY SET DEFAULT_CREDENTIAL = 'ADMIN.DEF_CRED_NAME'
```

   The `DEFAULT_CREDENTIAL` value cannot be an OCI resource principal, Azure service principal, Amazon Resource Name (ARN), or a Google service account. See Configure Policies and Roles to Access Resources for more information on resource principal based authentication.
4. Run Data Pump Export with the `dumpfile` parameter set to the URL for an existing bucket on your Cloud Object Storage (ending with a file name or a file name with a substitution variable, such as `exp%U.dmp`). For example:

```
expdp admin/password@db2022adb_high \
     filesize=5GB \
     dumpfile=default_credential:https://objectstorage.us-
ashburn-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp%U.dmp \
     parallel=16 \
     encryption_pwd_prompt=yes \
     logfile=export.log \
     directory=data_pump_dir
```

> **Note:**
>
> If during the export with `expdp` you use the `encryption_pwd_prompt=yes` parameter then use `encryption_pwd_prompt=yes` with your import and input the same password at the `impdp` prompt to decrypt the dump files (remember the password you supply with export). The maximum length of the encryption password is 128 bytes.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

The `default_credential` keyword in the `dumpfile` parameter is required.

For the best export performance use the `HIGH` database service for your export connection and set the `parallel` parameter to one quarter the number of ECPUs (.25 x **ECPU count**). If you are using OCPU compute model, set the parallel parameter to the number of OCPUs (1 x **OCPU count**).

For information on which database service name to connect to run Data Pump Export, see Manage Concurrency and Priorities on Autonomous Database.

For the dump file URL format for different Cloud Object Storage services, see DBMS_CLOUD URI Formats.

This example shows the recommended parameters for exporting from Autonomous Database. For these `expdp` parameters, note the following:

- The maximum `filesize` parameter value is 10000MB for Oracle Cloud Infrastructure Object Storage exports.
- The maximum `filesize` parameter value is 20GB for Oracle Cloud Infrastructure Object Storage Classic exports.
- If the specified `filesize` is too large, the export shows the error message:

  ```
  ORA-17500: ODM err:ODM HTTP Request Entity Too Large
  ```

- The `directory` parameter specifies the directory `data_pump_dir` for the specified log file, `export.log`. See Access Log Files for Data Pump Export for more information.

> **Note:**
>
> Oracle Data Pump divides each dump file part into smaller chunks for faster uploads. The Oracle Cloud Infrastructure Object Storage console shows multiple files for each dump file part that you export. The size of the actual dump files will be displayed as zero (0) and its related file chunks as 10mb or less. For example:
>
> ```
> exp01.dmp
> exp01.dmp_aaaaaa
> exp02.dmp
> exp02.dmp_aaaaaa
> ```
>
> Downloading the zero byte dump file from the Oracle Cloud Infrastructure console or using the Oracle Cloud Infrastructure CLI will not give you the full dump files. To download the full dump files from the Object Store, use a tool that supports Swift such as curl, and provide your user login and Swift auth token.
>
> ```
> curl -O -v -X GET -u 'user1@example.com:auth_token' \
>    https://swiftobjectstorage.us-ashburn-1.oraclecloud.com/v1/
> namespace-string/bucketname/exp01.dmp
> ```
>
> If you import a file with the `DBMS_CLOUD` procedures that support the `format` parameter `type` with the value '`datapump`', you only need to provide the primary file name. The procedures that support the '`datapump`' format type automatically discover and download the chunks.
>
> When you use `DBMS_CLOUD.DELETE_OBJECT`, the procedure automatically discovers and deletes the chunks when the procedure deletes the primary file.

5. Perform the required steps to use Oracle Data Pump import and clean up.

   See Download Dump Files, Run Data Pump Import, and Clean Up Object Store for details.

> **Note:**
>
> To perform a full export or to export objects that are owned by other users, you need the `DATAPUMP_CLOUD_EXP` role.

For detailed information on Oracle Data Pump Export parameters see *Oracle Database Utilities*.

Move Data to Object Store as Oracle Data Pump Files Using EXPORT_DATA

You can export data to Oracle Data Pump dump files by specifying a query.

With this export method you use the `DBMS_CLOUD.EXPORT_DATA` procedure to specify a query to select the data to export, as follows:

1. Connect to your database.
2. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adb_user@example.com',
    password => 'password'
  );
END;
/
```

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name.

See CREATE_CREDENTIAL Procedure for information about the `username` and `password` parameters for different object storage services.

3. Export data from Autonomous Database to your Cloud Object Store as Oracle Data Pump dump file(s) by calling `DBMS_CLOUD.EXPORT_DATA` with the `format` parameter `type` set to value `datapump`. For example:

```
BEGIN
 DBMS_CLOUD.EXPORT_DATA(
    credential_name =>'DEF_CRED_NAME',
    file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/exp01.dmp',
    format => json_object('type' value 'datapump'),
    query => 'SELECT warehouse_id, quantity FROM inventories'
 );
END;
/
```

The parameters are:

- `credential_name`: is the name of the credential created in the previous step.
- `file_uri_list`: is a comma delimited list of the export file(s). Use of wildcard and substitution characters is not supported in the `file_uri_list`.
- `format`: specifies the required `type` parameter with the value `datapump`, and optionally defines the options you can specify for the export with the `ORACLE_DATAPUMP` Access Driver.
- `query`: specifies a `SELECT` statement so that only the required data is exported. The query determines the contents of the dump file(s).

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

> **Note:**
>
> The `DBMS_CLOUD.EXPORT_DATA` procedure creates the dump file(s) that you specify in the `file_uri_list`. The procedure does not overwrite files. If a dump file in the `file_uri_list` exists, `DBMS_CLOUD.EXPORT_DATA` reports an error. `DBMS_CLOUD.EXPORT_DATA` does not create buckets.

For detailed information about the parameters, see EXPORT_DATA Procedure.

4. Perform the required steps to use Oracle Data Pump import and clean up. See Download Dump Files, Run Data Pump Import, and Clean Up Object Store for more details.

Notes for exporting data with `DBMS_CLOUD.EXPORT_DATA`:

- The dump files you create with `DBMS_CLOUD.EXPORT_DATA` cannot be imported using Oracle Data Pump `impdp`. Depending on the database, you can use these files as follows:
  - On an Autonomous Database, you can use the dump files with the `DBMS_CLOUD` procedures that support the `format` parameter `type` with the value '`datapump`'. You can import the dump files using `DBMS_CLOUD.COPY_DATA` or you can call `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` to create an external table.
  - On any other Oracle Database, such as Oracle Database 19c on-premise, you can import the dump files created with the procedure `DBMS_CLOUD.EXPORT_DATA` using the `ORACLE_DATAPUMP` access driver. See Unloading and Loading Data with the ORACLE_DATAPUMP Access Driver for more information.
- The number of dump files that `DBMS_CLOUD.EXPORT_DATA` generates is determined when the procedure runs. The number of dump files that are generated depends on the number of file names you provide in the `file_uri_list` parameter, as well as on the number of ECPUs available to the instance, the service level, and the size of the data.

  For example, if you use a 2 ECPU Autonomous Database instance or the `low` service, then a single dump file is exported with no parallelism, even if you provide multiple file names. If you use an 8 ECPU Autonomous Database instance with the `medium` or `high` service, then the jobs can run in parallel and multiple dump files are exported if you provide multiple file names.
- The `query` parameter value that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.

Download Dump Files, Run Data Pump Import, and Clean Up Object Store

If required, download the dump files from Cloud Object Store and use Oracle Data Pump Import to import the dump file set to the target database. Then perform any required clean up.

1. Download the dump files from Cloud Object Store.

> **Note:**
>
> This step is not needed if you are importing the data to an Autonomous Data Warehouse database, to an Autonomous Transaction Processing database, or to Autonomous JSON Database.

If you export directly to Object Store using Oracle Data Pump, as shown in Move Data with Data Pump Export to Object Store, then the dump files on Object Store show size 0. Oracle Data Pump divides each dump file part into smaller chunks for faster uploads. The Oracle Cloud Infrastructure Object Storage console shows multiple files for each dump file part that you export. The size of the actual dump files will be displayed as zero (0) and its related file chunks as 10mb or less. For example:

```
exp01.dmp
exp01.dmp_aaaaaa
exp02.dmp
exp02.dmp_aaaaaa
```

Downloading the zero byte dump file from the Oracle Cloud Infrastructure console or using the Oracle Cloud Infrastructure CLI will not give you the full dump files. To download the full dump files from the Object Store, use a tool that supports Swift such as curl, and provide your user login and Swift auth token.

```
curl -O -v -X GET -u 'user1@example.com:auth_token' \
   https://swiftobjectstorage.us-ashburn-1.oraclecloud.com/v1/namespace-
string/bucketname/exp01.dmp
```

The cURL command does not support wildcards or substitution characters in its URL. You need to use multiple cURL commands to download the dump file set from your Object Store. Alternatively, you can use a script that supports substitution characters to download all the dump files from your Object Store in a single command. See How To: Download all files from an export to object store job in Autonomous Database using cURL for an example.

2. Run Data Pump Import to import the dump file set to the target database.

   If you are importing the data to another Autonomous Database, see Import Data Using Oracle Data Pump on Autonomous Database.

3. Perform post import clean up tasks. If you are done importing the dump files to your target database then drop the bucket containing the data or remove the dump files from the Cloud Object Store bucket, and remove the dump files from the location where you downloaded the dump files to run Data Pump Import.

For detailed information on Oracle Data Pump Import parameters see *Oracle Database Utilities*.

Access Log Files for Data Pump Export

The log files for Data Pump Export operations are stored in the directory you specify with the data pump `directory` parameter.

To access the log file you need to move the log file to your Cloud Object Storage using the procedure `DBMS_CLOUD.PUT_OBJECT`. For example, the following PL/SQL block moves the file `export.log` to your Cloud Object Storage:

```
BEGIN
  DBMS_CLOUD.PUT_OBJECT(
    credential_name => 'DEF_CRED_NAME',
    object_uri => 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/import.log',
    directory_name  => 'DATA_PUMP_DIR',
```

```
    file_name => 'export.log');
END;
/
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.

See PUT_OBJECT Procedure for more information.

Oracle GoldenGate Capture for Oracle Autonomous Database

Using Oracle GoldenGate you can capture changes from an Oracle Autonomous Database and replicate to any target database or platform that Oracle GoldenGate supports, including another Oracle Autonomous Database.

Oracle GoldenGate Capture for Oracle Autonomous Database supports the following:

- Replication for different use cases: Report Offloading, Active-Active, Cloud to Cloud, and Cloud to on-premise.
- Inter-region and cross-region replication: Replicate data between different Oracle Cloud data centers around the world
- Replicate between targets: Replicate from an Autonomous Database to any target database or platform that Oracle GoldenGate supports, including to other Oracle Autonomous Database environments.

See Configure Replicat to Apply to an Autonomous Databases for more information.

Data Sharing

Share data on Autonomous Database

- Use Cloud Links for Read Only Data Access on Autonomous Database
  Cloud Links provide a cloud-based method to remotely access read only data on an Autonomous Database instance.
- Use Pre-Authenticated Request URLs for Read Only Data Access on Autonomous Database
- Share Data with Data Studio
  The Data Share tool enables you to share data and metadata with other users. The Data Share page displays information about the different types of shares available in the Oracle Autonomous Database.

Use Cloud Links for Read Only Data Access on Autonomous Database

Cloud Links provide a cloud-based method to remotely access read only data on an Autonomous Database instance.

- About Cloud Links on Autonomous Database
  Using Cloud Links a data owner registers a table or view for remote access for a selected audience, as defined by the data owner. The data is then made accessible to those with access granted at registration time. No further actions are required to set up Cloud Links, and whoever is supposed to see and access your data is able to discover and work with the data.

- Grant Cloud Links Access for Database Users
  The ADMIN user grants privileges to database users to register data sets. The ADMIN user also grants privileges to database users to access registered data sets.
- Register or Unregister a Data Set
  You can register a table or view you own as a registered data set. When you want to remove or replace the data set, you need to unregister it.
- Register or Unregister a Data Set in a Different Region
  You can use Cloud Links in multiple regions, where the source region contains the data set's source database and one or more remote regions contain refreshable clones of the source database.
- Register a Data Set with Authorization Required
  Optionally, when you register a data set, in addition to the scope you can specify that database level authorization is required to access a data set.
- Register a Data Set with Offload Targets for Data Set Access
  Optionally, when you register a data set you can offload access to the data set to one or more Autonomous Database instances that are refreshable clones.
- Use Cloud Links from a Read Only Autonomous Database Instance
  You can share Cloud Links when a data set resides on a Read-Only Autonomous Database instance.
- Find Data Sets and Use Cloud Links
  A user who is granted access to read Cloud Links can search for data sets available to an Autonomous Database instance and can access and use registered data sets with their queries.
- Revoke Cloud Links Access for Database Users
  The ADMIN user can revoke registration to disallow a user from registering data sets for remote access. The ADMIN user can also revoke privileges or database users to access registered data sets.
- Monitor and View Cloud Links Information
  Autonomous Database provides views that allow you to monitor and audit Cloud Links.
- Define a Virtual Private Database Policy to Secure a Registered Data Set
- Notes for Cloud Links
  Provides notes and restrictions for using Cloud Links.

About Cloud Links on Autonomous Database

Using Cloud Links a data owner registers a table or view for remote access for a selected audience, as defined by the data owner. The data is then made accessible to those with access granted at registration time. No further actions are required to set up Cloud Links, and whoever is supposed to see and access your data is able to discover and work with the data.

The Cloud Links implementation leverages Oracle Cloud Infrastructure access mechanisms to make data accessible within a specific scope. Scope indicates who can remotely access the data. Scope can be set to various levels, including to the region where the database resides, to individual tenancies, or to compartments. In addition, you can specify that authorization to access a data set is limited to one or more Autonomous Database instances.

By creating one or more cross-region refreshable clones from the source (data set owner's) Autonomous Database instance, you can use Cloud Links to share data across multiple regions.

Cloud Links greatly simplify the sharing of tables or views across Autonomous Database instances, as compared to traditional database linking mechanisms. With Cloud Links you can

discover data without requiring a complex database link setup. Autonomous Database provides transparent access using SQL, and implements privilege enforcement with the Cloud Links scope and by granting authorization to individual Autonomous Database instances.

Cloud Links introduce the concepts of regional namespaces and names for data that is made remotely accessible. This is similar to existing Oracle tables where there is a table, for example `"EMP"` that resides in a namespace (schema), for example `"LWARD"`. There can only be one `LWARD.EMP` in your database. Cloud Links provide a similar namespace and name on a regional level, that is not tied to a single database but applies across many Autonomous Database instances as specified with the scope and optionally with database authorization.

For example, you can register a data set under the namespace `FOREST` and for security purposes or for naming convenience, you can provide a namespace and a name other than the original schema and object names. In this example, `TREE_DATA` is the visible name of the registered data set and this name is not required to be the name of the source table. In addition to the namespace and the name, the `cloud$link` keyword indicates to the database that it must resolve the source as a Cloud Link.

To access a registered data set, include the namespace, the name, and the `cloud$link` keyword in the `FROM` clause of a `SELECT` statement:

```
SELECT county, species, height FROM FOREST.TREE_DATA@cloud$link;
```

Optionally, you can specify that access to a data set from one or more databases is offloaded to a refreshable clone. When a consumer Autonomous Database is listed in a data set's offload list, access to the data set is directed to the refreshable clone.

There are several concepts and terms to use when you work with Cloud Links:

- **Registered Data Set (Data Set)**: Identifies a table or view that has been enabled for remote access in an Autonomous Database. A registered data set also indicates who is allowed to access the data set (its scope). Data set registration defines a namespace and a name for use with Cloud Links. After data set registration, these values together specify the Fully Qualified Name (FQN) for remote access, and allow Cloud Links to manage accessibility for the data set.
- **Data Set Owner**: Specify a data set owner to provide a point of contact for questions about a data set.
- **Scope**: Specifies who and from where a user is allowed to access a registered data set. See Data Set Scope, Access Control, and Authorization for more details on scope.
- **OCID** (Oracle Cloud Identifier): Identifies a specific Tenancy, Compartment, or Database. The scope for a registered data set can be expressed in terms of OCIDs. See Resource Identifiers for more information.
- **Data Registration**: With data registration a user makes a table or view available for remote access, subject to the access restrictions imposed by scope, and optionally subject to an additional authorization step. You can allow remote access with Cloud Links on a table or view stored in the database or on data stored in Object Store.
- **Data Discovery**: A registered data set can be discovered using textual queries from the database. The discovery only shows a data set if there is a privilege to access the data set. You can search to find a registered data set by name or by description.
- **Data Describe**: Allows a user to retrieve a description or metadata for a table or view made available as a registered data set.
- **Offload Targets**: Optionally, you can specify one or more offload targets. Offload targets are refreshable clones that provide Cloud Links data sets to specified Autonomous

Database instances. By specifying an offload target, you can dedicate an Autonomous Database instance to provide data sets to separate production from development, for performance, to assure security, or for other reasons. See Use Refreshable Clones with Autonomous Database for more information.

Any access to a registered data set using Cloud Links is logged for audit purposes. The log includes the tenancy, compartment, or database that accessed the data set, the amount of data accessed, and additional information. The V$CLOUD_LINK_ACCESS_STATS and GV$CLOUD_LINK_ACCESS_STATS Views show audit information.

> **Note:**
>
> Cloud Links provide read only access to remote objects on an Autonomous Database instance. If you want to use database links with other Oracle databases or with non-Oracle databases, or if you want to use your remote data with DML operations, you need to use database links. See Use Database Links with Autonomous Database for more information.

Data Set Metadata and Audit Views

Each Autonomous Database instance provides views that expose data set metadata and that allow you to monitor and audit data usage. See Monitor and View Cloud Links Information for more information.

- Data Set Scope, Access Control, and Authorization

Data Set Scope, Access Control, and Authorization

Cloud Links leverage Oracle Cloud Infrastructure access mechanisms to make registered data sets accessible, and to protect your data with access restrictions.
Autonomous Database determines the accessibility of registered data sets as follows:

- The ADMIN user specifies a scope for a user that allows the user to register data sets based on the scope provided.
- A user who has been granted privileges to register data sets specifies a scope when they register a data set.
- Optionally, when you register a data set, a user who has been granted authorization privileges can specify that an authorization step required to access a data set. This authorization step is in addition to scope level access authorization.

Data Set Scope

The ADMIN sets a user's scope with `DBMS_CLOUD_LINK_ADMIN.GRANT_REGISTER` to be one of:

- `'MY$REGION'`
- `'MY$TENANCY'`
- `'MY$COMPARTMENT'`

A user's scope is hierarchical, meaning a user who is granted one of these scopes can allow access as follows:

- `MY$REGION`: A user can grant remote data access to other tenancies in the region of the Autonomous Database instance that is registering the data set. This is the least restrictive scope.

- `MY$TENANCY`: A user can grant remote data access to any resource, tenancy, compartment, or database in the tenancy of the Autonomous Database instance that is registering the data set. This scope is more restrictive than `MY$REGION` scope.
- `MY$COMPARTMENT`: A user can grant remote data access to any resource, compartment, or database in the compartment of the Autonomous Database instance that is registering the data set. This is the most restrictive scope you can set for a user with `GRANT_REGISTER`.

Next, the scope you set when you register a data set determines from where users are allowed to access the data set. The `DBMS_CLOUD_LINK.REGISTER scope` is a comma separated list consisting of one or more of the following:

- **Database OCID**: Access to the data set is allowed for the specific Autonomous Database instances identified by OCID.
- **Compartment OCID**: Access to the data set is allowed for databases in the compartments identified by compartment OCID.
- **Tenancy OCID**: Access to the data set is allowed for databases in the tenancies identified by tenancy OCID.
- **Region name**: Access to the data set is allowed for databases in the region identified by the named region. In all cases, Cloud Links access is limited to within a single region and is not cross-region.
- **`MY$COMPARTMENT`**: Access to the data set is allowed for databases in the same compartment as the data set owner.
- **`MY$TENANCY`**: Access to the data set is allowed for databases in the same tenancy as the data set owner.
- **`MY$REGION`**: Access to the data set is allowed for databases in the same region as the data set owner.

The scope values, `MY$REGION`, `MY$TENANCY`, and `MY$COMPARTMENT` are variables that act as convenience macros and resolve to OCIDs.

> **Note:**
>
> The scope you set when you register a data set is only honored when it matches or is more restrictive than the value set with `DBMS_CLOUD_LINK_ADMIN.GRANT_REGISTER`. For example, assume the ADMIN granted the scope `'MY$TENANCY'` with `GRANT_REGISTER`, and the user specifies `'MY$REGION'` when they register a data set with `DBMS_CLOUD_LINK.REGISTER`. In this case they would see an error such as the following:
>
> ```
> ORA-20001: Share privileges are not enabled for current user or it is
> enabled but not for scope MY$REGION
> ```

You can also use an additional access control mechanism based on a `SYS_CONTEXT` value. This mechanism uses the function `DBMS_CLOUD_LINK.GET_DATABASE_ID` that returns an identifier that is available as a `SYS_CONTEXT` value.

When you register a data set with `DBMS_CLOUD_LINK.REGISTER` you can use the `SYS_CONTEXT` value in Oracle Virtual Private Database (VPD) security policies to control database access to further restrict and control what specific data can be accessed by individual Autonomous Database instances.

See Define a Virtual Private Database Policy to Secure a Registered Data Set for more information on using VPD policies.

The database ID value is also available in the V$CLOUD_LINK_ACCESS_STATS and GV$CLOUD_LINK_ACCESS_STATS Views that tracks access statistics and audit information.

See Using Oracle Virtual Private Database to Control Data Access for more information.

Data Set Authorization

When you register a data set, if you have been granted authorization privileges you can specify that database OCID authorization is required to access a data set. To provide database OCID authorization for a data set, use the `DBMS_CLOUD_LINK.GRANT_AUTHORIZATION` procedure to specify the Autonomous Database instances that are authorized to access the data set. Before you run `DBMS_CLOUD_LINK.GRANT_AUTHORIZATION`, the ADMIN must authorize you to run this procedure with `DBMS_CLOUD_LINK_ADMIN.GRANT_AUTHORIZE`.

Data set registration with authorization required specifies database level access for a data set, in addition to the scope access control specified for the data set, as follows:

- Databases that are within the `SCOPE` specified and have been authorized with `DBMS_CLOUD_LINK.GRANT_AUTHORIZATION` can view rows from the data set.
- Any databases that are within the specified `SCOPE` but have not been authorized with `DBMS_CLOUD_LINK.GRANT_AUTHORIZATION` cannot view data set rows. In this case, consumers without authorization see the data set as empty.
- Databases that are not within the `SCOPE` specified see an error when attempting to access the data set.

Grant Cloud Links Access for Database Users

The ADMIN user grants privileges to database users to register data sets. The ADMIN user also grants privileges to database users to access registered data sets.

When the ADMIN user grants register privileges, they provide a scope that specifies the maximum scope that a user can provide when they register a data set (within the scope hierarchy). The valid `scope` values for use with `DBMS_CLOUD_LINK_ADMIN.GRANT_REGISTER` are:

- `'MY$REGION'`
- `'MY$TENANCY'`
- `'MY$COMPARTMENT'`

See Data Set Scope, Access Control, and Authorization for more information.

1. As the ADMIN user, allow a user to register data sets within a specified scope.

```
BEGIN
DBMS_CLOUD_LINK_ADMIN.GRANT_REGISTER(
   username => 'DB_USER1',
   scope    => 'MY$REGION');
END;
/
```

This specifies that the user `DB_USER1`, has privileges to register data sets within a specified scope, `'MY$REGION'`.

A user can query `SYS_CONTEXT('USERENV', 'CLOUD_LINK_REGISTER_ENABLED')` to check if they are enabled for registering data sets.

For example, the following query:

```
SELECT SYS_CONTEXT('USERENV', 'CLOUD_LINK_REGISTER_ENABLED') FROM DUAL;
```

Returns '`YES`' or '`NO`'.

See GRANT_REGISTER Procedure for more information.

2. As the ADMIN user, allow a user to access registered data sets.

```
EXEC DBMS_CLOUD_LINK_ADMIN.GRANT_READ('LWARD');
```

This allows `LWARD` to access registered data sets that are available to the Autonomous Database instance.

A user can query `SYS_CONTEXT('USERENV', 'CLOUD_LINK_READ_ENABLED')` to check if they are enabled for `READ` access to a data set.

For example, the following query:

```
SELECT SYS_CONTEXT('USERENV', 'CLOUD_LINK_READ_ENABLED') FROM DUAL;
```

Returns '`YES`' or '`NO`'.

See GRANT_READ Procedure for more information.

3. During data registration you can set the authorization required parameter to `TRUE`. When authorization required is `TRUE`, the ADMIN user must run `DBMS_CLOUD_LINK_ADMIN.GRANT_AUTHORIZE` to provide authorization to invoke `DBMS_CLOUD_LINK.GRANT_AUTHORIZATION`. Use `DBMS_CLOUD_LINK.GRANT_AUTHORIZATION` to specify authorization details.

   See Register a Data Set with Authorization Required for more information.

4. When the Autonomous Database instance has Database Vault enabled and the table or view to be registered with Cloud Links is protected by a realm, the owner of the table or view must be authorized to the realm as a realm-owner before registration.

```
BEGIN
     DBMS_MACADM.ADD_AUTH_TO_REALM(
           realm_name    => 'myrealm',
           grantee       => 'object_owner',
           auth_option   => dbms_macutl.g_realm_auth_owner);
END;
/
```

If the realm protecting the table or view is a mandatory realm, the Autonomous Database common schema named `C##DATA$SHARE`, which is used internally as the connecting schema, must be authorized to the realm as a realm-participant.

For example:

```
BEGIN
     DBMS_MACADM.ADD_AUTH_TO_REALM(
           realm_name    => 'myrealm',
```

```
            grantee      => 'C##DATA$SHARE',
            auth_option  => dbms_macutl.g_realm_auth_participant);
END;
/
```

See Use Oracle Database Vault with Autonomous Database for more information.

Notes for granting privileges to database users to register data sets:

- To register data sets or see and access remote data sets, you have to have granted the appropriate privilege for either registering with `DBMS_CLOUD_LINK_ADMIN.GRANT_REGISTER` or for reading data sets with `DBMS_CLOUD_LINK_ADMIN.GRANT_READ`.

  This is true for ADMIN user as well; however the ADMIN user can grant privileges to themself.
- The views `DBA_CLOUD_LINK_PRIVS` and `USER_CLOUD_LINK_PRIVS` provide information on user privileges. See Monitor and View Cloud Links Information for more information.
- A user can run the following query to check if they are enabled for authenticating registered, protected data sets:

  ```
  SELECT SYS_CONTEXT('USERENV', 'CLOUD_LINK_AUTH_ENABLED') FROM DUAL;
  ```

Register or Unregister a Data Set

You can register a table or view you own as a registered data set. When you want to remove or replace the data set, you need to unregister it.

Data set registration defines a namespace and a name for use with Cloud Links. After data set registration, these values together specify the Fully Qualified Name (FQN) for remote access, and allow Cloud Links to manage accessibility for the data set.

1. Obtain grant register privileges from the ADMIN user.

   See Grant Cloud Links Access for Database Users for more information.
2. Register one ore more data sets.

   For example, assuming there is a schema `CLOUDLINK` on your Autonomous Database instance, you can register two objects, `SALES_VIEW_AGG` and `SALES_ALL` and provide different `scope` parameters to determine how the objects are accessed.

   ```
   BEGIN
      DBMS_CLOUD_LINK.REGISTER(
       schema_name    => 'CLOUDLINK',
       schema_object  => 'SALES_VIEW_AGG',
       namespace      => 'REGIONAL_SALES',
       name           => 'SALES_AGG',
       description    => 'Aggregated regional sales information.',
       scope          => 'MY$TENANCY',
       auth_required  =>  FALSE,
       data_set_owner =>  'amit@example.com' );
   ```

```
END;
/


BEGIN
   DBMS_CLOUD_LINK.REGISTER(
    schema_name    => 'CLOUDLINK',
    schema_object  => 'SALES_ALL',
    namespace      => 'TRUSTED_COMPARTMENT',
    name           => 'SALES',
    description    => 'Trusted Compartment, only accessible within my
compartment. Early sales data.',
    scope          => 'MY$COMPARTMENT',
    auth_required  =>  FALSE,
    data_set_owner =>  'amit@example.com' );
END;
/
```

The parameters are:

- `schema_name`: is the schema name (the object owner).
- `schema_object`: is the object's name. Valid objects are:
  - Tables (including Heap, External or Hybrid)
  - Views
  - Materialized Views

  > **Note:**
  >
  > Other objects such as analytic views or synonyms are not supported.

- `namespace`: is the namespace you provide as a name for access with Cloud Links (one part of the Cloud Link FQN).

  A `NULL` value specifies a system-generated `namespace` value, unique to the Autonomous Database instance.
- `name`: is the name you provide for access with Cloud Links (one part of the Cloud Link FQN).
- `scope`: Specifies the scope. You can use one of the variables: `MY$REGION`, `MY$TENANCY`, or `MY$COMPARTMENT` or specify one or more OCIDs.

  See Data Set Scope, Access Control, and Authorization for more information.
- `auth_required`: A Boolean value that specifies if database level authorization is required for the data set, in addition to the scope access control. When this is set to `TRUE`, the data set enforces an additional authorization step. See Register a Data Set with Authorization Required for more information.
- `data_set_owner`: Text value specifies information about the individual responsible for the data set or a contact for questions about the data set. For example, you could supply an email address for the data set owner.

See REGISTER Procedure for more information.

After the registration completes, the scope for these two registered objects are different, based on the scope parameter:

- `MY$TENANCY`: Specifies tenancy level scope for `REGIONAL_SALES.SALES_AGG`.
- `MY$COMPARTMENT`: Specifies the more restrictive compartment level scope for my compartment within my tenancy for `TRUSTED_COMPARTMENT.SALES`.

If you want to revoke remote access to a registered data set, you can unregister the data set.

```
BEGIN
   DBMS_CLOUD_LINK.UNREGISTER(
    namespace      => 'TRUSTED_COMPARTMENT',
    name           => 'SALES');
END;
/
```

See UNREGISTER Procedure for more information.

- Notes for Registering or Unregistering a Data Set
  Provides notes for registering a data set with `DBMS_CLOUD_LINK.REGISTER` and unregistering a data set with `DBMS_CLOUD_LINK.UNREGISTER`.

Notes for Registering or Unregistering a Data Set

Provides notes for registering a data set with `DBMS_CLOUD_LINK.REGISTER` and unregistering a data set with `DBMS_CLOUD_LINK.UNREGISTER`.

- After you register an object, users may need to wait up to ten (10) minutes to access the object with Cloud Links.
- When you register a data set and you want consumers in a remote regions to access the data set, you must perform additional steps to make the data set available in a remote region. See Register or Unregister a Data Set in a Different Region for more information.
- To change the scope for an existing data set, you must first unregister the data set with `DBMS_CLOUD_LINK.UNREGISTER`. Then you need to register the data set with the new scope with `DBMS_CLOUD_LINK.REGISTER`. The total wait time for these two operations can be up to 20 minutes, including 10 minutes for the unregister and another 10 minutes for the register to be propagated and accessible through Cloud Links.

  This delay can also impact the visibility of the data set in the both the `DBA_CLOUD_LINK_REGISTRATIONS` and `DBA_CLOUD_LINK_ACCESS` views.
- You can register a table or view that resides in another user's schema when the you have `READ WITH GRANT OPTION` privileges for the table or view.
- Autonomous Database does not perform hierarchical validity checks at registration time and registrations that are outside the scope will never be visible or accessible.

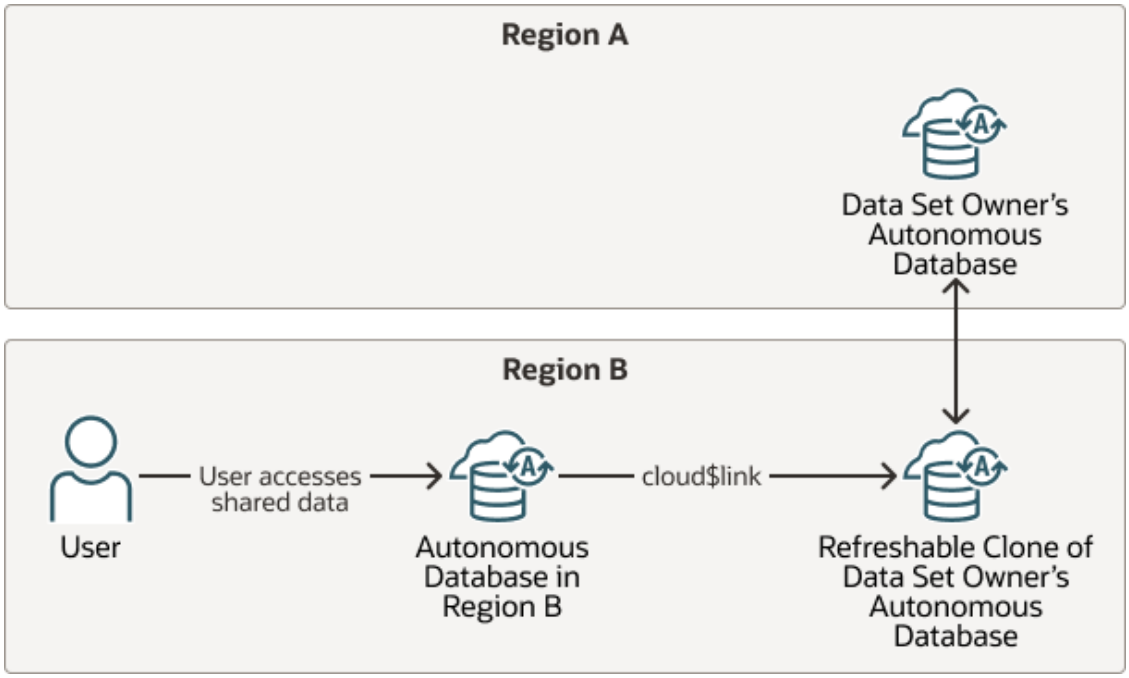
  For example, consider the following sequence:

  1. A user with scope `MY$COMPARTMENT` registers an object with a scope that specifies an individual database OCID.
  2. When a user requests access to the registered data set, Autonomous Database performs the check to see that the database OCID of the database where the request originates is in the OCID list specified with the `scope` when the data set was registered.
  3. After this, the `namespace.name` object will be discoverable, visible, and usable in the database where the request originated.

- `DBMS_CLOUD_LINK.UNREGISTER` may take up to ten (10) minutes to fully propagate, after which the data can longer be accessed remotely.

Register or Unregister a Data Set in a Different Region

You can use Cloud Links in multiple regions, where the source region contains the data set's source database and one or more remote regions contain refreshable clones of the source database.



To use Cloud Links with a data set in a different region:

1. Create a cross-region Refreshable Clone of the source database in a remote region.

   See Create a Cross Tenancy or Cross-Region Refreshable Clone for information on adding a cross-region refreshable clone.

2. On the source database, register the data set.

   See Register or Unregister a Data Set for more information.

3. Refresh the refreshable clone.

   See Refresh a Refreshable Clone on Autonomous Database for more information.

4. On the remote Refreshable Clone register the data set using the same arguments you used to register the data set in the source region.

   For example, assuming there is a schema `CLOUDLINK` on your Autonomous Database instance, after you the register `SALES_ALL` on the source database, register `SALES_ALL` on the refreshable clone:

```
BEGIN
   DBMS_CLOUD_LINK.REGISTER(
    schema_name     => 'CLOUDLINK',
    schema_object   => 'SALES_ALL',
    namespace       => 'TRUSTED_COMPARTMENT',
    name            => 'SALES',
    description     => 'Trusted Compartment, only accessible within my
```

```
compartment. Early sales data.',
    scope          => 'MY$COMPARTMENT',
    auth_required  =>  FALSE,
    data_set_owner =>  'amit@example.com' );
END;
/
```

The parameters are:

- `schema_name`: is the schema name (the object owner).
- `schema_object`: is the object's name. Valid objects are:
  - Tables (including Heap, External or Hybrid)
  - Views
  - Materialized Views

> **Note:**
>
> Other objects such as analytic views or synonyms are not supported.

- `namespace`: is the namespace you provide as a name for access with Cloud Links (one part of the Cloud Link FQN).

  A `NULL` value specifies a system-generated `namespace` value, unique to the Autonomous Database instance.
- `name`: is the name you provide for access with Cloud Links (one part of the Cloud Link FQN).
- `scope`: Specifies the scope. You can use one of the variables: `MY$REGION`, `MY$TENANCY`, or `MY$COMPARTMENT` or specify one or more OCIDs.

  See Data Set Scope, Access Control, and Authorization for more information.
- `auth_required`: A Boolean value that specifies if database level authorization is required for the data set, in addition to the scope access control. When this is set to `TRUE`, the data set enforces an additional authorization step. See Register a Data Set with Authorization Required for more information.
- `data_set_owner`: Text value specifies information about the individual responsible for the data set or a contact for questions about the data set. For example, you could supply an email address for the data set owner.

See REGISTER Procedure for more information.

After the registration completes on the refreshable clone, the scope for the registered object is `MY$COMPARTMENT`: Specifies the more restrictive compartment level scope for my compartment within my tenancy for `TRUSTED_COMPARTMENT.SALES`.

You can unregister a remote data set only in the remote regions, or in both the remote regions and in the source region:

To unregister a data set in a remote region and disable remote access to the data set:

1. On the refreshable clone, unregister the data set.

For example:

```
BEGIN
   DBMS_CLOUD_LINK.UNREGISTER(
    namespace       => 'TRUSTED_COMPARTMENT',
    name            => 'SALES');
END;
/
```

2. Refresh the refreshable clone.

   See Refresh a Refreshable Clone on Autonomous Database for more information.

To unregister a data set on the source database and unregister the data set on remote region refreshable clones:

1. On the remote refreshable clone if there is only one, or on multiple refreshable clones in remote regions if there are more than one, unregister the data set.

   For example:

```
BEGIN
   DBMS_CLOUD_LINK.UNREGISTER(
    namespace       => 'TRUSTED_COMPARTMENT',
    name            => 'SALES');
END;
/
```

2. On the source database, unregister the data set.

   See Register or Unregister a Data Set for more information.

3. Refresh the refreshable clones.

   See Refresh a Refreshable Clone on Autonomous Database for more information.

- Notes for Registering or Unregistering a Data Set in a Remote Region
  Provides notes for registering a data set in a remote region.

Notes for Registering or Unregistering a Data Set in a Remote Region

Provides notes for registering a data set in a remote region.

- When you register a data set on a refreshable clone in a remote region, the invocation of `DBMS_CLOUD_LINK.REGISTER` on the remote region clone must use the same parameters with the same values as on the source database, with the exception of the `offload_targets` parameter.

  For example, when you run `DBMS_CLOUD_LINK.REGISTER` with scope set to `MY$COMPARTMENT` on the source Autonomous Database instance, run the procedure again on the cross-region refreshable clone with the same scope parameter value (`MY$COMPARTMENT`).

- If you specify the `offload_targets` parameter with `DBMS_CLOUD_LINK.REGISTER` on source, you should omit this parameter when you register the data set on the refreshable clone.

- After you register an object, users may need to wait up to ten (10) minutes to access the object with Cloud Links.

- The following actions require that you refresh the refreshable clone:

  – If you add a VPD policy to the data set in the source, you must refresh the refreshable clone.

- If you perform a grant or a revoke for the data set on the source database, you must refresh the refreshable clone.

See Refresh a Refreshable Clone on Autonomous Database for more information.

Register a Data Set with Authorization Required

Optionally, when you register a data set, in addition to the scope you can specify that database level authorization is required to access a data set.

Compared to the previous example where you set `auth_required` to `FALSE`, in this example you set `auth_required` to `TRUE`. When `auth_required` is `TRUE`, additional steps are required to specify one or more databases from which access to the data set is authorized.

> **Note:**
>
> You can only grant authorization, as shown in these steps, if you are authorized. The ADMIN grants authorization privileges with `DBMS_CLOUD_LINK_ADMIN.GRANT_AUTHORIZE`.

1. Use `DBMS_CLOUD_LINK.REGISTER` to register a data with authorization required.

   Assuming there is a schema `CLOUDLINK` on your Autonomous Database instance and you register the object `SALES_VIEW_AGG` and set `auth_required` to `TRUE`, then in addition to defining the scope, you must preform additional steps to determine how the object is accessed.

```
BEGIN
   DBMS_CLOUD_LINK.REGISTER(
    schema_name    => 'CLOUDLINK',
    schema_object  => 'SALES_VIEW_AGG',
    namespace      => 'REGIONAL_SALES',
    name           => 'SALES_AGG',
    description    => 'Aggregated regional sales information.',
    scope          => 'MY$TENANCY',
    auth_required  =>  TRUE,
    data_set_owner =>  'amit@example.com' );
END;
/
```

   The parameters are:

   - `schema_name`: is the schema name (the object owner).
   - `schema_object`: is the object's name. Valid objects are:
     - Tables (including Heap, External or Hybrid)
     - Views
     - Materialized Views

     > **Note:**
     >
     > Other objects such as analytic views or synonyms are not supported.

- `namespace`: is the namespace you provide as a name for access with Cloud Links (one part of the Cloud Link FQN).

  A `NULL` value specifies a system-generated `namespace` value, unique to the Autonomous Database instance.

- `name`: is the name you provide for access with Cloud Links (one part of the Cloud Link FQN).
- `scope`: Specifies the scope. You can use one of the variables: `MY$REGION`, `MY$TENANCY`, or `MY$COMPARTMENT` or specify one or more OCIDs.

  See Data Set Scope, Access Control, and Authorization for more information.

- `auth_required`: A Boolean value that specifies if database level authorization is required for the data set, in addition to the scope access control. When this is set to `TRUE`, the data set enforces an additional authorization step.
- `data_set_owner`: Text value specifies information about the individual responsible for the data set or a contact for questions about the data set. For example, you could supply an email address for the data set owner.

2. Obtain the database ID for the database you want to grant authorization to (to allow the database to access to the data set).

   On the system you want to grant access the data set:

   ```
   SELECT DBMS_CLOUD_LINK.GET_DATABASE_ID FROM DUAL:
   ```

3. Use the database ID you obtained to grant authorization to a specified data set.

   You can only grant authorization, and run `DBMS_CLOUD_LINK.GRANT_AUTHORIZATION`, as shown in this step, if you are authorized. The ADMIN grants authorization with `DBMS_CLOUD_LINK_ADMIN.GRANT_AUTHORIZE`.

   ```
   BEGIN
      DBMS_CLOUD_LINK.GRANT_AUTHORIZATION(
       database_id    => '120xxxxxxx8506029999',
       namespace      => 'TRUSTED_COMPARTMENT',
       name           => 'SALES');
   END;
   /
   ```

   Perform these steps multiple times if you want to authorize additional databases.

If you want to revoke authorization for a database:

```
BEGIN
   DBMS_CLOUD_LINK.REVOKE_AUTHORIZATION(
    database_id    => '120xxxxxxx8506029999',
    namespace      => 'TRUSTED_COMPARTMENT',
    name           => 'SALES');
END;
/
```

See the following for more information:

- REGISTER Procedure
- GRANT_AUTHORIZE Procedure

- REVOKE_AUTHORIZATION Procedure

Register a Data Set with Offload Targets for Data Set Access

Optionally, when you register a data set you can offload access to the data set to one or more Autonomous Database instances that are refreshable clones.

Use the optional `offload_targets` parameter with `DBMS_CLOUD_LINK.REGISTER` to specify that access is offloaded to refreshable clones. The source database for each refreshable clone is the Autonomous Database instance where you register the data set (data publisher).

The `offload_targets` value is a JSON document that defines one or more `CLOUD_LINK_DATABASE_ID` and `OFFLOAD_TARGET` key value pairs:

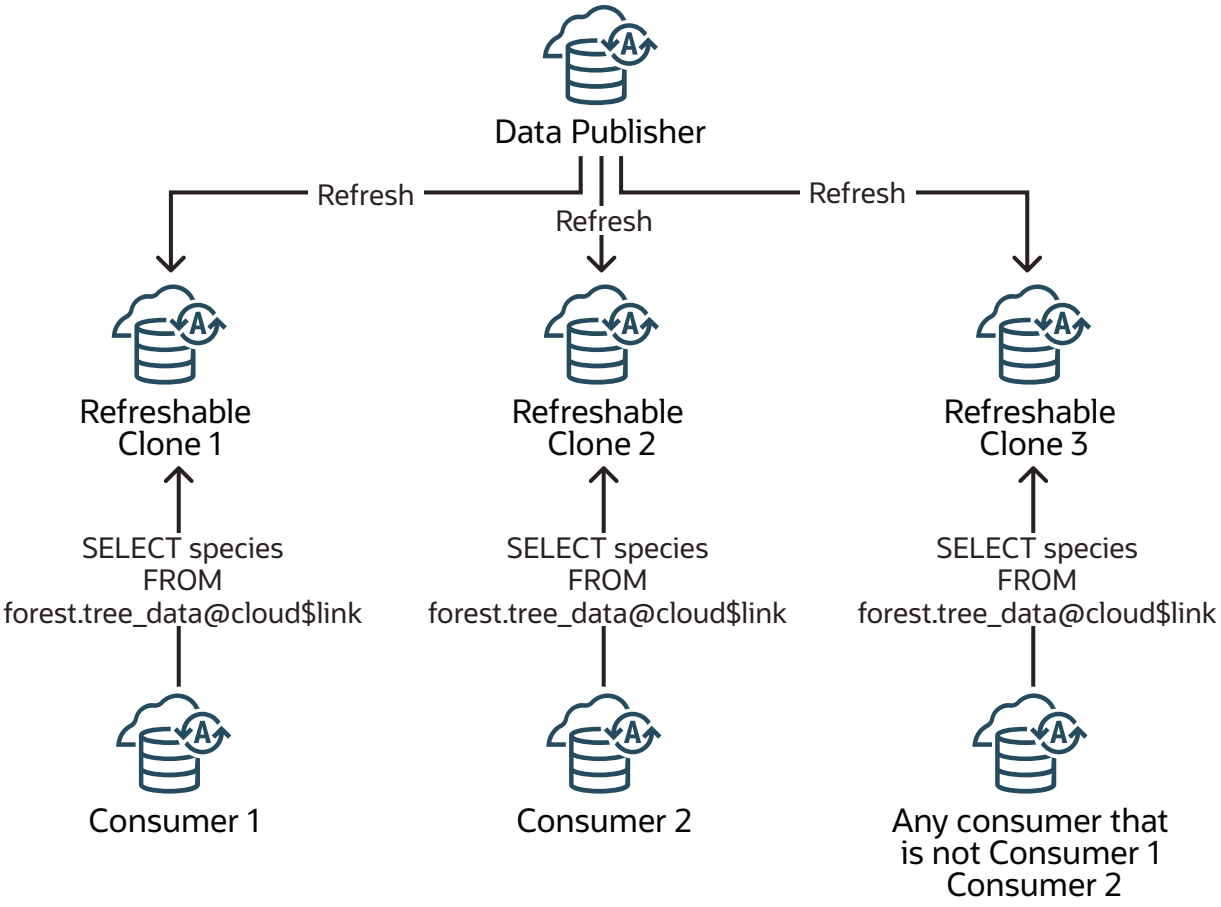
- `CLOUD_LINK_DATABASE_ID` is one of:
  - A database ID: This specifies a database ID for the data set consumer whose request is offloaded to the corresponding refreshable clone specified with the `OFFLOAD_TARGET` value.

    Obtain the database ID by running `DBMS_CLOUD_LINK.GET_DATABASE_ID`. See GET_DATABASE_ID Function for more information.
  - `ANY`: This specifies that any data set consumer's request is offloaded to the corresponding offload target. A consumer's data set request will be routed to the corresponding offload target.

    If you specify `ANY` without specifying database IDs, all data set requests from consumers are offloaded to the refreshable clone specified with the `OFFLOAD_TARGET` value.

    If you specify both database IDs and `ANY`, data set requests from consumers that do not match a database ID are offloaded to the refreshable clone specified with the `OFFLOAD_TARGET` value.
- `OFFLOAD_TARGET` is the OCID for an Autonomous Database instance that is a refreshable clone.

The following figure illustrates using offload targets.



OFFLOAD_TARGETS JSON	
CLOUD LINK DATABASE ID	**OFFLOAD TARGET**
Consumer 1's Database ID	Refreshable Clone 1
Consumer 2's Database ID	Refreshable Clone 2
ANY	Refreshable Clone 3

When a data set consumer requests access to a data set that you register with `offload_targets` and the Autonomous Database instance's database ID matches the value specified in `CLOUD_LINK_DATABASE_ID`, access is offloaded to the refreshable clone identified with `OFFLOAD_TARGET` in the supplied JSON.

For example, the following shows a JSON sample with three `OFFLOAD_TARGET`/ `CLOUD_LINK_DATABASE_ID` value pairs:

```
{
  "OFFLOAD_TARGETS": [
    {
      "CLOUD_LINK_DATABASE_ID": "34xxxxx69708978",
      "OFFLOAD_TARGET":
"ocid1.autonomousdatabase.oc1..xxxxx3pv6wkcr4jqae5f44n2b2m2yt2j6rx32uzr4h25vqs
```

```
tifsfabc"
    },
    {
      "CLOUD_LINK_DATABASE_ID": "34xxxxx89898978",
      "OFFLOAD_TARGET":
"ocid1.autonomousdatabase.oc1..xxxxx3pv6wkcr4jqae5f44n2b2m2yt2j6rx32uzr4h25vqs
tifsfdef"
    },
    {
      "CLOUD_LINK_DATABASE_ID": "34xxxxx4755680",
      "OFFLOAD_TARGET":
"ocid1.autonomousdatabase.oc1..xxxxx3pv6wkcr4jqae5f44n2b2m2yt2j6rx32uzr4h25vqs
tifsfghi"
    }
  ]
}
```

When a data set consumer requests access to a data set that you register with `offload_targets` that includes the `ANY` keyword, access is offloaded to the refreshable clone identified with `OFFLOAD_TARGET` in the supplied JSON (except for requests from consumers that have a matching database ID entry in the supplied JSON).

For example, the following shows a JSON sample with one explicit `OFFLOAD_TARGET`/ `CLOUD_LINK_DATABASE_ID` value pair, and one `ANY` value with a corresponding `OFFLOAD_TARGET`:

```
{
  "OFFLOAD_TARGETS": [
    {
      "CLOUD_LINK_DATABASE_ID": "ANY",
      "OFFLOAD_TARGET":
"ocid1.autonomousdatabase.oc1..xxxxx3pv6wkcr4jqae5f44n2b2m2yt2j6rx32uzr4h25vqs
tifsfdef"
    },
    {
      "CLOUD_LINK_DATABASE_ID": "34xxxxx4755680",
      "OFFLOAD_TARGET":
"ocid1.autonomousdatabase.oc1..xxxxx3pv6wkcr4jqae5f44n2b2m2yt2j6rx32uzr4h25vqs
tifsfghi"
    }
  ]
}
```

To register a data set and specify offload targets, do the following:

1. Obtain the OCID for one or more refreshable clones where you want to offload data set access. Refreshable clone OCIDs are available on the Oracle Cloud Infrastructure Console on a refreshable clone.

> **Note:**
>
> It can take up to 10 minutes after you create a refreshable clone for the refreshable clone to be visible as an offload target. This means you may need to wait up to 10 minutes after you create a refreshable clone for the refreshable clone to be available for Cloud Links offload registration.

2. Obtain the database ID for one or more Autonomous Database instances that you want to access the data set using data provided by the refreshable clone.

   On the system you want to access the data set from a refreshable clone, run the command:

   ```
   SELECT DBMS_CLOUD_LINK.GET_DATABASE_ID FROM DUAL:
   ```

3. Register a data set and specify the `offload_targets` parameter.

   For example, assuming there is a schema `CLOUDLINK` on your Autonomous Database instance, you can register `SALES_VIEW_AGG` and specify the refreshable clones that provide access to the data set:

   ```
   BEGIN
      DBMS_CLOUD_LINK.REGISTER(
       schema_name     => 'CLOUDLINK',
       schema_object   => 'SALES_VIEW_AGG',
       namespace       => 'REGIONAL_SALES',
       name            => 'SALES_AGG',
       description     => 'Aggregated regional sales information.',
       scope           => 'MY$TENANCY',
       auth_required   =>  FALSE,
       data_set_owner  =>  'amit@example.com',
       offload_targets => '{
     "OFFLOAD_TARGETS": [
       {
         "CLOUD_LINK_DATABASE_ID": "34xxxx754755680",
         "OFFLOAD_TARGET":
   "ocid1.autonomousdatabase.oc1..xxxxxaaba3pv6wkcr4jqae5f44n2b2m2yt2j6rx32uzr
   4h25vqstifsfghi"
       }
     ]
   }');
   END;
   /
   ```

   The parameters are:

   - `schema_name`: is the schema name (the object owner).
   - `schema_object`: is the object's name. Valid objects are:
     - Tables (including Heap, External or Hybrid)
     - Views
     - Materialized Views

> **Note:**
>
> Other objects such as analytic views or synonyms are not supported.

- `namespace`: is the namespace you provide as a name for access with Cloud Links (one part of the Cloud Link FQN).

  A `NULL` value specifies a system-generated `namespace` value, unique to the Autonomous Database instance.

- `name`: is the name you provide for access with Cloud Links (one part of the Cloud Link FQN).

- `scope`: Specifies the scope. You can use one of the variables: `MY$REGION`, `MY$TENANCY`, or `MY$COMPARTMENT` or specify one or more OCIDs.

  See Data Set Scope, Access Control, and Authorization for more information.

- `auth_required`: A Boolean value that specifies if database level authorization is required for the data set, in addition to the scope access control. When this is set to `TRUE`, the data set enforces an additional authorization step. See Register a Data Set with Authorization Required for more information.

- `data_set_owner`: Text value specifies information about the individual responsible for the data set or a contact for questions about the data set. For example, you could supply an email address for the data set owner.

- `offload_targets`: Specifies one or more Autonomous Database OCIDs of refreshable clones where access to data sets is offloaded, from the Autonomous Database where the data set is registered.

  For each data set consumer, one of the following can match to offload the request to a refreshable clone:

  – When there is matching the value of the specified `cloud_link_database_id`, that is the values matches the consumer's database ID, access is offloaded to the refreshable clone identified by the OCID specified in `offload_target`.

  – When the `ANY` keyword is specified, if there is not a matching the value of a specified `cloud_link_database_id`, access is offloaded to the refreshable clone identified with the ANY entry by the OCID specified in the corresponding `offload_target`.

See REGISTER Procedure for more information.

See Use Refreshable Clones with Autonomous Database for more information.

Use Cloud Links from a Read Only Autonomous Database Instance

You can share Cloud Links when a data set resides on a Read-Only Autonomous Database instance.

To share Cloud Links from an Autonomous Database instance that is in Read-Only mode:

1. Change the database mode to Read/Write mode.

   See Change Autonomous Database Operation Mode to Read/Write Read-Only or Restricted for more information.

2. With the database in Read/Write mode, perform the steps to register a data set.

   a. Grant Cloud Links Access for Database Users

b. Register or Unregister a Data Set

3. After you register one or more data sets, change the database to read-only mode.

   See Change Autonomous Database Operation Mode to Read/Write Read-Only or Restricted for more information.

Find Data Sets and Use Cloud Links

A user who is granted access to read Cloud Links can search for data sets available to an Autonomous Database instance and can access and use registered data sets with their queries.

After the ADMIN user runs `GRANT_READ`, a user can search for and use Cloud Links.

1. Find the available data sets on your Autonomous Database instance.

   For example, search for all data sets that contain the string, "TREE":

```
DECLARE
   result CLOB DEFAULT NULL;
BEGIN
   DBMS_CLOUD_LINK.FIND('TREE', result);
    DBMS_OUTPUT.PUT_LINE(result);
END;
/

[{"name":"TREE_DATA","namespace":"FOREST","description":"Urban tree data
including county, species and height"}]
```

   The parameters are:

   - `search_string`: The string to search for. The search string is not case sensitive.
   - `search_result`: A JSON document that includes the namespace, name, and description values for the data set.

   See FIND Procedure for more information.

   The `DBMS_CLOUD_LINK.FIND` procedure provides the FQN you can use with Cloud Links. In this case, `FOREST.TREE_DATA`.

2. Use the view `DBA_CLOUD_LINK_ACCESS` to list available data sets:

```
SELECT * FROM DBA_CLOUD_LINK_ACCESS;

NAMESPACE            NAME
---------            --------------
FOREST               TREE_DATA
REGIONAL_SALES       SALES_AGG
TRUSTED_COMPARTMENT  SALES
```

3. Use the procedure `DBMS_CLOUD_LINK.DESCRIBE` to find out more details about a registered data set.

```
SELECT DBMS_CLOUD_LINK.DESCRIBE('FOREST','TREE_DATA') FROM DUAL;

DBMS_CLOUD_LINK.DESCRIBE('FOREST','TREE_DATA')
```

```
-------------------------------------------------
Urban tree data including county, species and height
```

4. Use the registered data set in a query.

```
SELECT * FROM FOREST.TREE_DATA@cloud$link;
```

> **Note:**
>
> It can take up to 10 minutes after you register a data set with `DBMS_CLOUD_LINK.REGISTER` for the data set to be visible and accessible.

Revoke Cloud Links Access for Database Users

The ADMIN user can revoke registration to disallow a user from registering data sets for remote access. The ADMIN user can also revoke privileges or database users to access registered data sets.

1. As the ADMIN user, revoke a user's privileges to register data sets.

   For example:

   ```
   EXEC DBMS_CLOUD_LINK_ADMIN.REVOKE_REGISTER('DB_USER1');
   ```

   This revokes the privileges to register data sets for the user, `DB_USER1`.

   Running `DBMS_CLOUD_LINK_ADMIN.REVOKE_REGISTER` does not affect data sets that are already registered. Use `DBMS_CLOUD_LINK.UNREGISTER` to remove access for a registered data set.

   See REVOKE_REGISTER Procedure and UNREGISTER Procedure for more information.

2. As the ADMIN user, revoke access to registered data sets.

   For example:

   ```
   EXEC DBMS_CLOUD_LINK_ADMIN.REVOKE_READ('LWARD');
   ```

   This revokes access to Cloud Links data sets for the user `LWARD`.

   See REVOKE_READ Procedure for more information.

Monitor and View Cloud Links Information

Autonomous Database provides views that allow you to monitor and audit Cloud Links.

View	Description
V$CLOUD_LINK_ACCESS_STATS and GV$CLOUD_LINK_ACCESS_STATS Views	Use to track access to each registered data set on the Autonomous Database instance. These views track elapsed time, CPU time, the number of rows retrieved, and additional information about registered data sets. You can use the information in these views to audit Cloud Links data set access and usage.

View	Description
DBA_CLOUD_LINK_REGISTRATIONS and ALL_CLOUD_LINK_REGISTRATIONS Views	Use to list details of the registered data sets on an Autonomous Database instance.
DBA_CLOUD_LINK_ACCESS and ALL_CLOUD_LINK_ACCESS Views	Use to retrieve details of registered data sets a database is allowed to access.
DBA_CLOUD_LINK_AUTHORIZATIONS View	Provides information about which databases are authorized to access which data sets. This applies to data sets where the `auth_required` parameter is `TRUE`.
DBA_CLOUD_LINK_PRIVS and USER_CLOUD_LINK_PRIVS Views	Provides information on the Cloud Link specific privileges, `REGISTER`, `READ`, or `AUTHORIZE`, granted to all users or to the current user.

Define a Virtual Private Database Policy to Secure a Registered Data Set

By defining Virtual Private Database (VPD) policies for a registered data set, you can provide fine-grained Cloud Link access control so that only a subset of data (rows) is visible for specific remote sites.

Oracle Virtual Private Database (VPD) is a security feature that lets you control data access dynamically at row level for users and applications by applying filters on the same data set.

Any user who is granted access to read Cloud Links can access and use registered data sets if they are within the scope specified when the data set is registered, and if the extra authorization required parameter is set for the data set, the access is from an authorized database. Each remote access is done in the context of the remote Autonomous Database instance accessing the registered data set (on the database where the data set was registered).

You use the function `DBMS_CLOUD_LINK.GET_DATABASE_ID` on the remote system to get the database's unique ID. By defining a VPD policy on the database that registered a data set, you now can use the remote database's identifier as `SYS_CONTEXT` rule to provide more fine-grained control. You can define rules for remote databases accessing your registered data set and limit access beyond what is possible by specifying a Cloud Link scope.

Consider an example where `REGIONAL_SALES.SALES_AGG` is made available at the tenancy level. If you want to restrict access to all databases except one specific database, only allowing full access to the specified database, you can add a VPD policy on the registered data set.

For example:

1. Obtain the unique Cloud Link database ID for the Autonomous Database instance where you want to provide full access.

```
DECLARE
     DB_ID NUMBER;
 BEGIN
     DB_ID := DBMS_CLOUD_LINK.GET_DATABASE_ID;
     DBMS_OUTPUT.PUT_LINE('Database ID:' || DB_ID);
 END;
 /
```

2. Create VPD policy on the database that registered the data set by only allowing full access to the one specific database whose identifier you got on the database in question in Step 1.

```
CREATE OR REPLACE FUNCTION vpd_policy_sales(
     owner IN VARCHAR2,
     object IN VARCHAR2)
     RETURN VARCHAR2 IS
BEGIN
  IF SYS_CONTEXT('USERENV', 'CLOUD_LINK_DATABASE_ID')  <>
'12121212163948244901' THEN
    RETURN 'time_id >= trunc(sysdate,''year'')';
  ELSE
    RETURN '';
  END IF;
END;
/
```

   See DBMS_RLS for more information.

3. Register the VPD policy for your registered data set to limit full access to only the database identified in step 1.

```
BEGIN
  DBMS_RLS.ADD_POLICY(object_schema => 'CLOUDLINK',
            object_name => 'SALES_VIEW_AGG',
            policy_name => 'THIS_YEAR',
            function_schema => 'ADMIN',
            policy_function => 'VPD_POLICY_SALES',
            statement_types => 'SELECT',
            policy_type => DBMS_RLS.SHARED_CONTEXT_SENSITIVE);
END;
/
```

   See DBMS_RLS for more information.

See Using Oracle Virtual Private Database to Control Data Access for more information.

Notes for Cloud Links

Provides notes and restrictions for using Cloud Links.

- There is a limit of 4096 on the number of data sets you can register.

  Each Autonomous Database instance can register no more than 4096 data sets. This limit applies to every Autonomous Database instance, regardless of the number of ECPUs (OCPUs if your database uses OCPUs) or the storage size of the instance. The limit is a fixed value and setting the **ECPU count** to a larger value does not allow you to register more data sets.

- You can register an object in another schema if you have `READ WITH GRANT OPTION` privilege on the object.
- To register data sets or see and access remote data sets, you have to have granted the appropriate privilege for either registering or reading data sets. This is true for ADMIN as well; however ADMIN can grant this privilege to themself.

- To use `DBMS_CLOUD_LINK.REGISTER`, you must have execute privilege on the `DBMS_CLOUD_LINK` package, in addition to the register privilege assigned with `DBMS_CLOUD_LINK_ADMIN.GRANT_REGISTER`. Only the ADMIN user and schemas with `PDB_DBA` have this privilege by default.
- If you drop and re-create a table that was registered, you need to re-register the table for remote access.
- Only the ADMIN user and users with the role `PDB_DBA` are privileged to access the following views:
  - `DBA_CLOUD_LINK_ACCESS`
  - `DBA_CLOUD_LINK_REGISTRATIONS`
  - `DBA_CLOUD_LINK_AUTHORIZATIONS`
  - `DBA_CLOUD_LINK_PRIVS`

  See DBMS_CLOUD_LINK Views for more information.
- Accessing registered, remote data requires the remote database to be open. If the remote database is closed or in restricted mode, you will not be able to access the data and an Oracle error will be returned.
- There is a limit of a maximum of four open database links per session. Exceeding this limit can lead to `ORA-02020` or `ORA-12545`.
- If result cache is enabled, like the default behavior in Autonomous Database with **Data Warehouse** workload, you need to ensure that result cache is not used when realtime data is required.
- When you update your license type from Free to Paid, you must re-register Cloud Links data sets. See Update Always Free Instance to Paid with Autonomous Database for more information.
- Cloud Links remote connectivity uses the `LOW` service and this cannot be changed. You can see the remote connections as user `C##DATA$SHARE` in `V$SESSION` and the Cloud Links views V$CLOUD_LINK_ACCESS_STATS and GV$CLOUD_LINK_ACCESS_STATS Views provide more details about remote connections.
- All interfaces are case-sensitive unless otherwise noted, as follows:
  - Anything you enter that exists in the database, for example user names and table names, case is significant and must be entered with uppercase letters.
  - Predefined variables, for example the predefined scope values must be entered in uppercase letters.
  - Anything that you specify for the setup of Cloud Links, for example namespaces or names of tables in a namespace, must be specified as entered. For example, if you define a namespace as `trees`, you must enclose the namespace in double quotes, as `"trees"`, when accessing it with SQL.
- You can share Cloud Links when a data set resides on an Autonomous Database instance in Read-Only mode. See Use Cloud Links from a Read Only Autonomous Database Instance for more information.
- It can take up to 10 minutes after you create a refreshable clone for the refreshable clone to be visible as an offload target. This means you may need to wait up to 10 minutes after you create a refreshable clone for the refreshable clone to be available for Cloud Links offload registration.

  See Register a Data Set with Offload Targets for Data Set Access and Create a Refreshable Clone for an Autonomous Database Instance for more information.

Use Pre-Authenticated Request URLs for Read Only Data Access on Autonomous Database

You can generate and manage Pre-Authenticated Request (PAR) URLs for data on Autonomous Database.

Using a PAR URL allows you to easily retrieve data from the database, without requiring you to provide additional information other than the PAR URL to access the data. Any user can access the data by supplying the PAR URL in a browser or using a REST client, subject to security controls enforced by the database.

- About Pre-Authenticated Request (PAR) URLs on Autonomous Database
  Depending on how you generate a Pre-Authenticated Request (PAR) URL, a PAR URL provides access to data in tables or views or by running a SQL query.
- Generate a PAR URL for a Table or a View
  Shows you the steps to generate a PAR URL that you can use to share access for a schema object (table or view).
- Generate a PAR URL with a Select Statement
  Shows you the steps to generate a PAR URL that provides access to data using a SQL query statement.
- Define a Virtual Private Database Policy to Secure PAR URL Data
- List PAR URLs
  You can list the active PAR URLs that you generated on an Autonomous Database instance and the ADMIN user can list all active PAR URLs.
- Use a PAR URL to Access Data
  PAR URL data is retrieved and returned in JSON format and is paginated.
- Invalidate PAR URLs
  At any time a user with appropriate privileges can invalidate a PAR URL.
- Monitor and View PAR URL Usage
  Autonomous Database provides views that allow you to monitor PAR URL usage.
- Notes for Using PAR URLs to Share Data
  Provides notes for using PAR URLs with Autonomous Database.

About Pre-Authenticated Request (PAR) URLs on Autonomous Database

Depending on how you generate a Pre-Authenticated Request (PAR) URL, a PAR URL provides access to data in tables or views or by running a SQL query.

When you generate a PAR URL you specify an expiration, either as an expiration time, for example set the PAR URL to expire after 120 minutes, or as an expiration count, for example the PAR URL expires after the PAR URL is used 10 times.

PAR URLs provide the following:

- **Public Access**: Using a PAR URL a data recipient on the public internet can access data when the data resides on an Autonomous Database instance in a private subnet.
- **Expiration**: A data provider specifies expiration for a PAR URL, meaning the PAR URL has a limited time before it expires (up to a maximum of 90 Days).
- **Expiration Use Count Limits**: A data provider can specify a limit on how many times a recipient can use a PAR URL to access data.

- **Endpoint Transparency**: A data provider is able to hide the Autonomous Database name so that it is not visible in a PAR URL.

PAR URL Use Cases

Generating and providing PAR URLs supports the following use cases:

Use Case	Description
Within Organization Collaboration	You can use PAR URLs for emergency data access. In situations where a rapid response is needed, such as during a critical incident investigation, you provide a PAR URL to allow immediate and temporary access to specific data without the need to create new database accounts or modify existing permissions.
BB (Business to Business) Applications	A business partner can easily access data. Using a PAR URL, a business can provide a business partner with a simple way to access data or reports. This can eliminate the need for manual report generation and email distribution.
Third-party Audits and Reviews	When an external auditor or reviewer requires access to specific data for a limited time, a PAR URL can give them the access they need without compromising the overall security of the database.
Data as a Product (Digital Commerce)	Vendors can grant limited or single-use access to purchased content or data using a PAR URL. Once accessed, the URL expires, protecting the product's exclusivity and ensuring efficient, secure delivery.

Security Best Practices for PAR URLs

Following are some best practices for generating and using PAR URLs:

- **Set a Short Expiration Time**: A PAR URL should only be valid for the minimum time required. The shorter the validity period, the lower the risk if the PAR URL is compromised.
- **PAR Invalidation**: Invalidate a PAR URL immediately when it is no longer required.
- **Use Appropriate Permissions**: A PAR URL runs using the privileges granted to the database user who generates the PAR URL. The user that generates a PAR URL should have the minimum privileges required for providing access to the data.
- **Content Security**: To mitigate risk of sharing unintended dynamic data:
  - Create a view on top of the data that you want to share in a PAR URL, and monitor that the view definition is up to date.
  - As needed, create a VPD policy when you generate a PAR URL. You can use VPD policies to restrict the rows visible to the PAR URL users.
- **Load Monitoring**: Monitor the PAR URL query load using PerfHub and SQL monitoring.

  Enable **Compute auto scaling** and make sure that the CPU count is appropriately sized for the data set size and PAR URL query load.

Generate a PAR URL for a Table or a View

Shows you the steps to generate a PAR URL that you can use to share access for a schema object (table or view).

When a PAR URL runs it uses the privileges granted to the database user who generates the PAR URL. The user that generates a PAR URL should have the minimum privileges required for providing access to the data. To maintain security, Oracle recommends that you do not run `DBMS_DATA_ACCESS.GET_PREAUTHENTICATED_URL` as the ADMIN user.

To use a PAR URL to provide access to data as a schema object (table or view):

1. Identify the table or view that you want to share.

   If there are restrictions on the data you want to make available, use the `application_user_id` parameter when you generate the PAR URL and create a VPD policy to restrict the data that you expose. See Define a Virtual Private Database Policy to Secure PAR URL Data for more information.

2. Run `DBMS_DATA_ACCESS.GET_PREAUTHENTICATED_URL` to generate the PAR URL.

```
DECLARE
   status CLOB;
   BEGIN
      DBMS_DATA_ACCESS.GET_PREAUTHENTICATED_URL(
          schema_name => 'ADMIN',
          schema_object_name    => 'TREE_DATA',
          expiration_minutes    => 360,
          result                => status);
       dbms_output.put_line(status);
    END;
/
```

   The `DBMS_DATA_ACCESS.GET_PREAUTHENTICATED_URL` procedure does not accept bind variables.

   See GET_PREAUTHENTICATED_URL Procedure for more information.

3. Check the result.

   In this example `status` contains the result that includes information about the PAR URL.

```
{
"status":"SUCCESS",
"id":"Vd1Px7QWASdqDbnndiuwTA_example",
"preauth_url":"https://dataaccess.adb.us-
ashburn-1.oraclecloudapps.com/adb/p/Vdtv..._example_wxd0PN/data",
"expiration_ts":"2023-12-04T23:51:35.334Z"
}
```

   You can also use `DBMS_DATA_ACCESS.LIST_ACTIVE_URLS` to show PAR URLs. See List PAR URLs for more information.

Generate a PAR URL with a Select Statement

Shows you the steps to generate a PAR URL that provides access to data using a SQL query statement.

When a PAR URL runs it uses the privileges granted to the database user who generates the PAR URL. The user that generates a PAR URL should have the minimum privileges required for providing access to the data. To maintain security, Oracle recommends that you do not run `DBMS_DATA_ACCESS.GET_PREAUTHENTICATED_URL` as the ADMIN user.

To use a PAR URL to provide to access to data as an arbitrary SQL query statement:

1. Identify the table or view that contains the information you want to share, as well as the `SELECT` statement on the table or view that you want to use.

If there are restrictions on the data you want to make available, use the `application_user_id` parameter when you generate the PAR URL and create a VPD policy to restrict the data that you expose. See Define a Virtual Private Database Policy to Secure PAR URL Data for more information.

2. Run `DBMS_DATA_ACCESS.GET_PREAUTHENTICATED_URL` to generate the PAR URL.

```
DECLARE
   status CLOB;
   BEGIN
      DBMS_DATA_ACCESS.GET_PREAUTHENTICATED_URL(
          sql_statement        => 'SELECT species, height FROM TREE_DATA',
          expiration_count     => 10,
          result               => status);
       dbms_output.put_line(status);
    END;
/
```

The `DBMS_DATA_ACCESS.GET_PREAUTHENTICATED_URL` procedure does not accept bind variables in the SQL statement you provide with `sql_statement`.

> **Note:**
>
> The `sql_statement` value must be a `SELECT` statement.

This `expiration_count` parameter specifies the PAR URL expires and is invalidated after 10 uses and without an `expiration_time` specified, the expiration time is set to the default 90 days.

See GET_PREAUTHENTICATED_URL Procedure for more information.

3. Check the result.

In this example `status` contains the result that includes the PAR URL.

```
{
   "status":"SUCCESS",
   "id":"xA6iRGexmapleARc_zz",
   "preauth_url":"https://dataaccess.adb.us-
ashburn-1.oraclecloudapps.com/adb/p/XA2_example_m4s/data",
   "expiration_ts":"2024-05-16T17:01:08.226Z",
   "expiration_count":10
}
```

You can use `DBMS_DATA_ACCESS.LIST_ACTIVE_URLS` to show the PAR URLs. See List PAR URLs for more information.

Define a Virtual Private Database Policy to Secure PAR URL Data

By defining Oracle Virtual Private Database (VPD) policies for data that you share with a PAR URL, you can provide fine-grained access control so that only a subset of data, rows, is visible for a specific PAR URL.

Oracle Virtual Private Database (VPD) is a security feature that lets you control data access dynamically at row level for users and applications by applying filters on the same data set. When a PAR URL is accessed, the value of `application_user_id` specified during PAR URL generation is available through `sys_context('DATA_ACCESS_CONTEXT$', 'USER_IDENTITY')`. You can define VPD Policies that make use of the value of this Application Context to restrict the data, rows, visible to the application user.

Any user who is granted access to read data with a PAR URL can access and use the data (either a table, a view, or the data provided with a select statement). By defining a VPD policy on the database that generated a PAR URL, you can use the `application_user_id` value in a SYS_CONTEXT rule to provide more fine-grained control. Consider an example where data is made available with a PAR URL. If you want to restrict access to some of the data you can add a VPD policy.

For example:

1. Obtain the `application_user_id` value that you specified when you generated the PAR URL.
2. Create VPD policy on the database where you generated the PAR URL.

   ```
   CREATE OR REPLACE FUNCTION limit_sal (v_schema IN VARCHAR2, v_objname IN
   VARCHAR2)
     RETURN VARCHAR2 authid current_user AS
   BEGIN
        RETURN 'employee_id = SYS_CONTEXT(''DATA_ACCESS_CONTEXT$'',
   ''USER_IDENTITY'')';
   END;
   ```

   See DBMS_RLS for more information.
3. Register the VPD policy.

   ```
   BEGIN
      DBMS_RLS.ADD_POLICY(
           object_schema        => 'HR',
           object_name          => 'EMPLOYEE',
           policy_name          => 'POL',
           policy_function      => 'LIMIT_SAL');
   END;
   /
   ```

   See DBMS_RLS for more information.

See Using Oracle Virtual Private Database to Control Data Access for more information.

List PAR URLs

You can list the active PAR URLs that you generated on an Autonomous Database instance and the ADMIN user can list all active PAR URLs.

Run `DBMS_DATA_ACCESS.LIST_ACTIVE_URLS` to list the active PAR URLs. For example:

```
DECLARE
    result CLOB;
    BEGIN
       result := DBMS_DATA_ACCESS.LIST_ACTIVE_URLS;
       dbms_output.put_line(result);
    END;
/
```

See LIST_ACTIVE_URLS Function for more information.

> **Note:**
>
> The behavior of `DBMS_DATA_ACCESS.LIST_ACTIVE_URLS` is dependent on the invoker. If the invoker is ADMIN or any user with `PDB_DBA` role, the function lists all active PAR URLs, regardless of the user who generated the PAR URL. If the invoker is not the ADMIN user and not a user with `PDB_DBA` role, the list includes only the active PAR URLs generated by the invoker.

Use a PAR URL to Access Data

PAR URL data is retrieved and returned in JSON format and is paginated.

You can access the data using a PAR URL with a browser or using any REST client. The data returned is paginated to allow you to access a maximum of 100 records at a time, with the total data size in the response limited to 1 MB. You can provide the `limit` query parameter to limit the number of records fetched. PAR URL data retrieval is blocked if PAR URL authentication fails or if the requested PAR URL has expired.

For example, use a PAR URL:

```
curl https://dataaccess.adb.us-chicago-1.oraclecloudapps.com/adb/p/K6XExample/
data
```

The PAR URL response includes links for any previous or next pages, when the data includes more than one page. This allows you to navigate in either direction while fetching data. The JSON also includes a `self` link that points to the current page, as well as a `hasMore` attribute that indicates if there is more data available to fetch.

The following is the response format:

```
{
    "items": [],                <-- Array of records from database
    "hasMore": true OR false,   <-- Indicates if there are more records to
fetch or not
    "limit": Number,            <-- Indicates number of records in the page.
```

```
Maximum allowed number is 100.
    "offset": Number,           <-- Offset indicating the start of the
current page
    "count": Number,            <-- Count of records in the current page
    "links": [
        {
            "rel": "self",
            "href": "{Link to preauth url for the current page}"
        },
        {
            "rel": "previous",
            "href": "{Link to preauth url for the previous page}"
        },
        {
            "rel": "next",
            "href": "{Link to preauth url for the next page}"
        }
    ]
}
```

For example, the following is a sample response from a PAR URL (with newlines added for clarity):

```
{"items":[
{"COUNTY":"Main","SPECIES":"Alder","HEIGHT":45},
{"COUNTY":"First","SPECIES":"Chestnut","HEIGHT":51},
{"COUNTY":"Main","SPECIES":"Hemlock","HEIGHT":17},
{"COUNTY":"Main","SPECIES":"Douglas-fir","HEIGHT":34},
{"COUNTY":"First","SPECIES":"Larch","HEIGHT":12},
{"COUNTY":"Main","SPECIES":"Cedar","HEIGHT":21},
{"COUNTY":"First","SPECIES":"Douglas-fir","HEIGHT":10},
{"COUNTY":"Main","SPECIES":"Yew","HEIGHT":11},
{"COUNTY":"First","SPECIES":"Willow","HEIGHT":17},
{"COUNTY":"Main","SPECIES":"Pine","HEIGHT":29},
{"COUNTY":"First","SPECIES":"Pine","HEIGHT":16},
{"COUNTY":"First","SPECIES":"Spruce","HEIGHT":6},
{"COUNTY":"Main","SPECIES":"Spruce","HEIGHT":8},
{"COUNTY":"First","SPECIES":"Hawthorn","HEIGHT":19},
{"COUNTY":"First","SPECIES":"Maple","HEIGHT":16},
{"COUNTY":"Main","SPECIES":"Aspen","HEIGHT":35},
{"COUNTY":"First","SPECIES":"Larch","HEIGHT":27},
{"COUNTY":"First","SPECIES":"Cherry","HEIGHT":20},
{"COUNTY":"Main","SPECIES":"Pine","HEIGHT":37},
{"COUNTY":"Main","SPECIES":"Redwood","HEIGHT":78},
{"COUNTY":"Main","SPECIES":"Alder","HEIGHT":45},
{"COUNTY":"First","SPECIES":"Chestnut","HEIGHT":51},
{"COUNTY":"Main","SPECIES":"Hemlock","HEIGHT":17},
{"COUNTY":"Main","SPECIES":"Douglas-fir","HEIGHT":34},
{"COUNTY":"First","SPECIES":"Larch","HEIGHT":12},
{"COUNTY":"Main","SPECIES":"Cedar","HEIGHT":21},
{"COUNTY":"First","SPECIES":"Douglas-fir","HEIGHT":10},
{"COUNTY":"Main","SPECIES":"Redwood","HEIGHT":78}],

"hasMore":false,
"limit":100,
```

```
"offset":0,
"count":30,
"links":
[
{"rel":"self",
"href":"https://dataaccess.adb.us-ashburn-1.oraclecloudapps.com/adb/p/
F5Sn..._example/data"}
]}
```

Invalidate PAR URLs

At any time a user with appropriate privileges can invalidate a PAR URL.

To invalidate a PAR URL, you need the PAR URL `id`. Use `DBMS_DATA_ACCESS.LIST_ACTIVE_URLS` to list each PAR URLs and its associated `id`.

Use `DBMS_DATA_ACCESS.INVALIDATE_URL` to invalidate a PAR URL. For example:

```
DECLARE
    status CLOB;
    BEGIN
       DBMS_DATA_ACCESS.INVALIDATE_URL(
        id => 'Vd1Px7QWASdqDbnndiuwTAyyEstv82PCHqS_example',
        result => status);
       dbms_output.put_line(status);
    END;
/
```

See INVALIDATE_URL Procedure for more information.

Monitor and View PAR URL Usage

Autonomous Database provides views that allow you to monitor PAR URL usage.

Views	Description
V$DATA_ACCESS_URL_STATS and GV$DATA_ACCESS_URL_STATS Views	These views track PAR URL usage, including elapsed time, CPU time, and additional information.

Notes for Using PAR URLs to Share Data

Provides notes for using PAR URLs with Autonomous Database.

- The `DBMS_DATA_ACCESS.GET_PREAUTHENTICATED_URL` procedure does not accept bind variables, whether the PAR URL is on a SQL statement or on a specific object.
- When you run the `DBMS_DATA_ACCESS.GET_PREAUTHENTICATED_URL` procedure with the `sql_statement` parameter, the SQL statement must be a `SELECT` statement.
- There is a limit of 128 active PAR URLs on an Autonomous Database instance.

Share Data with Data Studio

The Data Share tool enables you to share data and metadata with other users. The Data Share page displays information about the different types of shares available in the Oracle Autonomous Database.

See The Data Share Tool for more information.

- The Data Share Tool
  The Data Share tool allows you to share Oracle data and metadata with other databases and non-database tools. The Data Share page displays information about the different types of shares available in the Oracle Autonomous Database.

The Data Share Tool

The Data Share tool allows you to share Oracle data and metadata with other databases and non-database tools. The Data Share page displays information about the different types of shares available in the Oracle Autonomous Database.

The following topics describe the Data Share page and how to use it:

- Overview of Share
- Access and Enable the Data Share Tool
- Provide Share
- Consume Share
- Data Share Limitations
- FAQs on the Data Share Tool
- Overview of the Data Share Tool
  Oracle Autonomous Database enables you to create shares using the share tool.
- Access and Enable the Data Share Tool
  To share or consume data using the Data Share tool, navigate to the Data Studio, launch the Data Share tool, and enable sharing.
- Provide Share
  Use the Provide Share page to view the number of shares of the database, and the number of share recipients, search for a share entity, search for a recipient, create a new share, and create a new recipient.
- Consume Share
  Once the providers share the objects, there are a few steps the recipients need to follow to consume the share.
- Data Share Limitations on Autonomous Database
  This section summarizes the limitations of Data Share (both Versioned and Live Share) included in the Oracle Autonomous Database.
- FAQs on the Data Share Tool
  The following are frequently asked questions on the Data Share tool.

Overview of the Data Share Tool

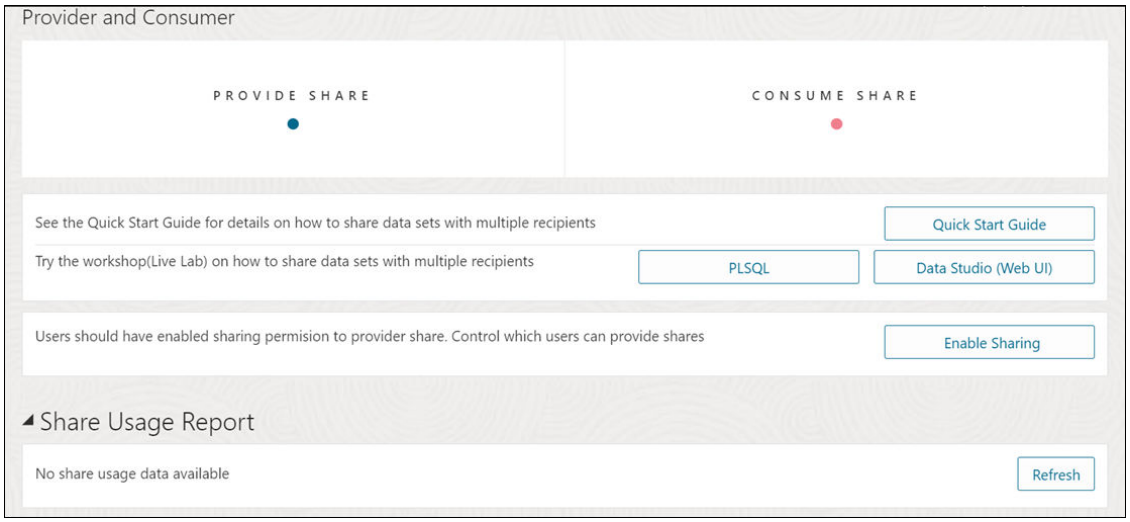
Oracle Autonomous Database enables you to create shares using the share tool.

Sharing objects requires two steps. The Provider provides data share for access and the Consumer role consumes (or receives) access to the published shares. The provider creates a share with the objects to share in the desired cloud object location. The provider also adds the recipient. The recipient accepts and receives the configured shared objects from the provider for consumption.

> **Note:**
>
> You must have the correct privileges to create or consume a data share. In case the Data Share card is disabled, click on the tool tip and follow the steps for the administrator to grant you the required privilege.

Select the **Data Share** menu from the Data Studio suite in the Database Actions homepage to access this tool. This opens the **Data Share** homepage. It consists of widgets that enable you to provide and consume share objects.

> **Note:**
>
> This is the homepage you view after you have enabled sharing and set Provider identification details.



> **Note:**
>
> If you do not see the Data Share tool card then your database user is missing the required DWROLE role.

Click Quick Start Guide to familiarize yourself with the Data Share tool.

Click PLSQL or Data Studio (Web UI) to try Data Sharing with PL/SQL or Data Studio without creating an account on Oracle Cloud tenancy.

Click **Enable Sharing** to grant sharing permission to you as a provider. See Access and Enable the Data Share Tool for more details.

The widgets are defined in the following sections:

- Provide Share
- Consume Share

Share Terminology

Provider: The Autonomous Database Serverless enables the provider to share existing objects. The share can contain a single table, a set of related tables, or a set of tables with some logical grouping. It could be a person, an institution, or a software system that shares the objects.

Example: An institution, such as NASA, that makes a data set available via data.gov.

Recipient: A Share recipient is an entity that associates an individual, an institution or a software system that receives a share from a provider. A recipient can have access to multiple shares. If you remove a recipient, that recipient loses access to all shares it could previously access.

Example: An external system, such as Microsoft Power BI, that supports the Delta Sharing REST API.

Share: A Share is a named entity in the provider's instance. It can be a group of datasets shared as a single entity.

Example: A SALES table that needs to be shared within an organization.

Overview of Providers and Recipients

A Data Share is the logical container that contains objects (such as tables) that share recipients will get access to a share and all tables within this share. A Data Share also implements security mechanisms on a high object level which simplifies the authorization for a set of individual objects. A provider creates and publishes share of a versioned type. The recipient is given access to a share. The provider can modify shares (both data and metadata) after the provider publishes the share to the recipients.

Use Case of Data Share

A marketing agency can share sales information with multiple interested parties. The Data Analysis tool analyses the data, generates insights and then the application shares the information with interested parties.

How the Data Share tool works?

Data is made accessible by the data sharing provider (that is an Oracle Autonomous Database) to the data sharing recipient at query time in parquet format for a versioned share. A live share uses cloud links and can only be consumed in an Oracle database The provider can only share data which they have access to when they log into an autonomous database instance.

As a data provider, you create a share and select other additional entities to share. The Oracle Data Sharing for general recipients is based on the open delta sharing standard protocol, providing a simple REST-based API to share data in parquet format. For near real-time access

to shared data, customers can use Live Shares accessed using the consumer's ADB-S instance.

The Autonomous Database Serverless Versioned Sharing protocol works as follows:

- The provider creates and publishes a share that can be shared with one or multiple recipients. Every recipient will get a personal activation link to download their own JSON profile with the necessary information to access their share.
- The versioned share recipient registers with the share server by entering the URL for the end point along with a client ID, secret key and a bearer token.
- The versioned share recipient retrieves data from the share by calling the /shares/../tables/../query endpoint to obtain a list of URLs. The recipient then sends a GET request on these URLs to obtain the parquet files.

The Autonomous Database Serverless Live Sharing protocol works as follows:

- For a Live Share, the intended recipient will copy the sharing ID from the consumer page and publish the share that can be shared with recipients. This is the case when provider shares to only one PDB.
- A provider can share also share to **ALL_REGIONS**, **ALL_TENANCY**, or **ALL_COMPARTMENTS**.

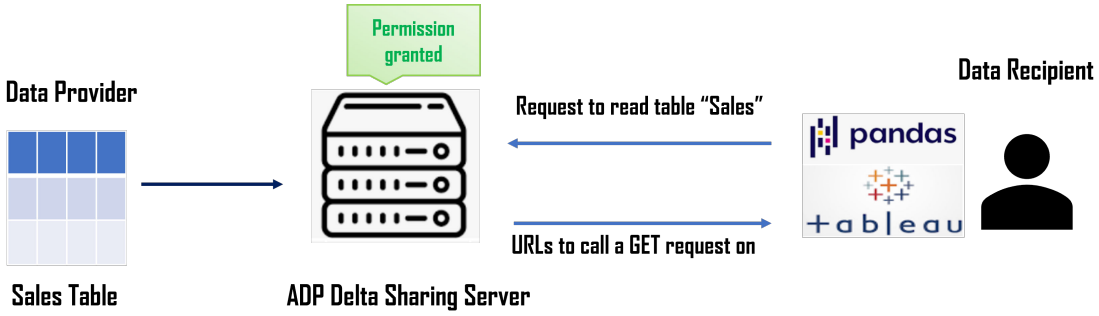
Features of Autonomous Database Serverless Share

With Autonomous Database Share you can:

- Share objects easily across Autonomous Databases and all tools or APIs that support the open delta sharing protocol.
- Share versioned data with many recipients without data replication for all recipients.
- Establish secure and centrally managed data sharing and collaboration within and across organizations.

Share Architecture

The following diagram is a generalized flow diagram of the architecture of the Data Share.



Prerequisites for Share Providers

Here are some prerequisites for a share provider to use the share tool:

- For a versioned share, you must have read and write access to a bucket to store or cache your shares.
- The schema you wish to use to create and publish shares must be enabled by an `ADMIN` user.

Prerequisites for Share Recipients

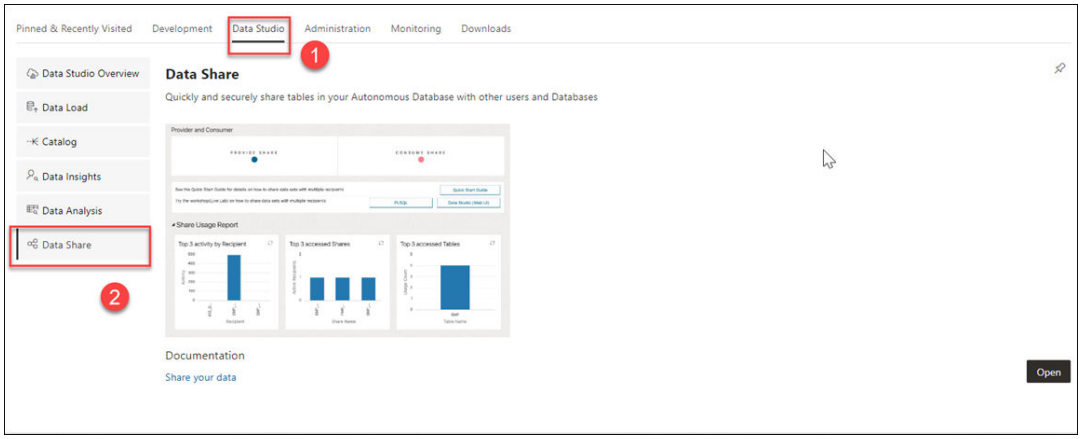
The share recipient must have a valid email address a provider can use to register the recipient to use the share tool. Oracle Data Share allows to share the information about a recipient's activation link by email.

Access and Enable the Data Share Tool

To share or consume data using the Data Share tool, navigate to the Data Studio, launch the Data Share tool, and enable sharing.

Follow these steps:

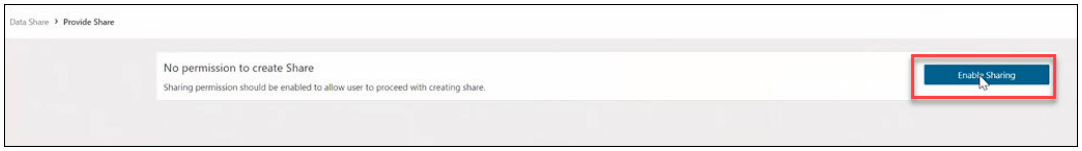
1. Navigate to the **Launchpad** on your Database Actions instance. See Accessing Database Actions for further instructions.
2. Click the **Data Studio** tab and select the **Data Share** pane.



3. Clicking the Data Share pane opens a Provider and Consumer page where you can view **Provide Share** and **Consume Share** widgets.

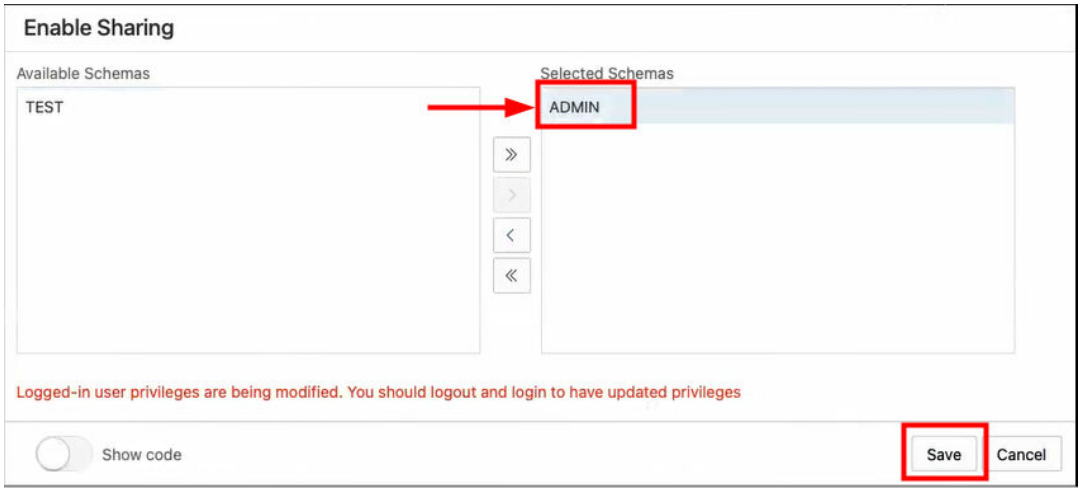
Click **Provide Share**.

4. On the **Provide Share** page, click **Enable Sharing**.

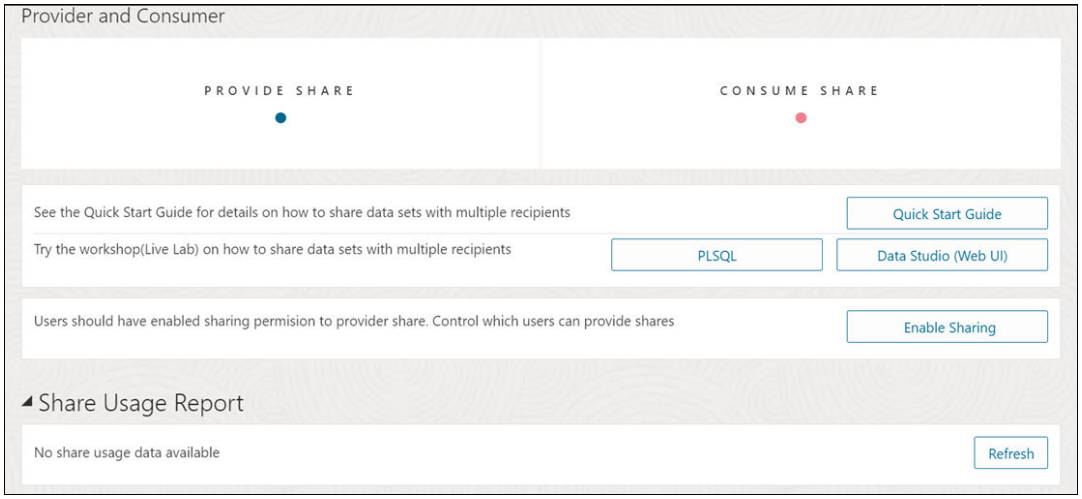


Clicking the **Enable Share** icon takes you to the Enable Sharing screen where you must select the user schema you want to enable and move it from the Available Schemas column to the Selected Schemas column.

5. Click **Save**.



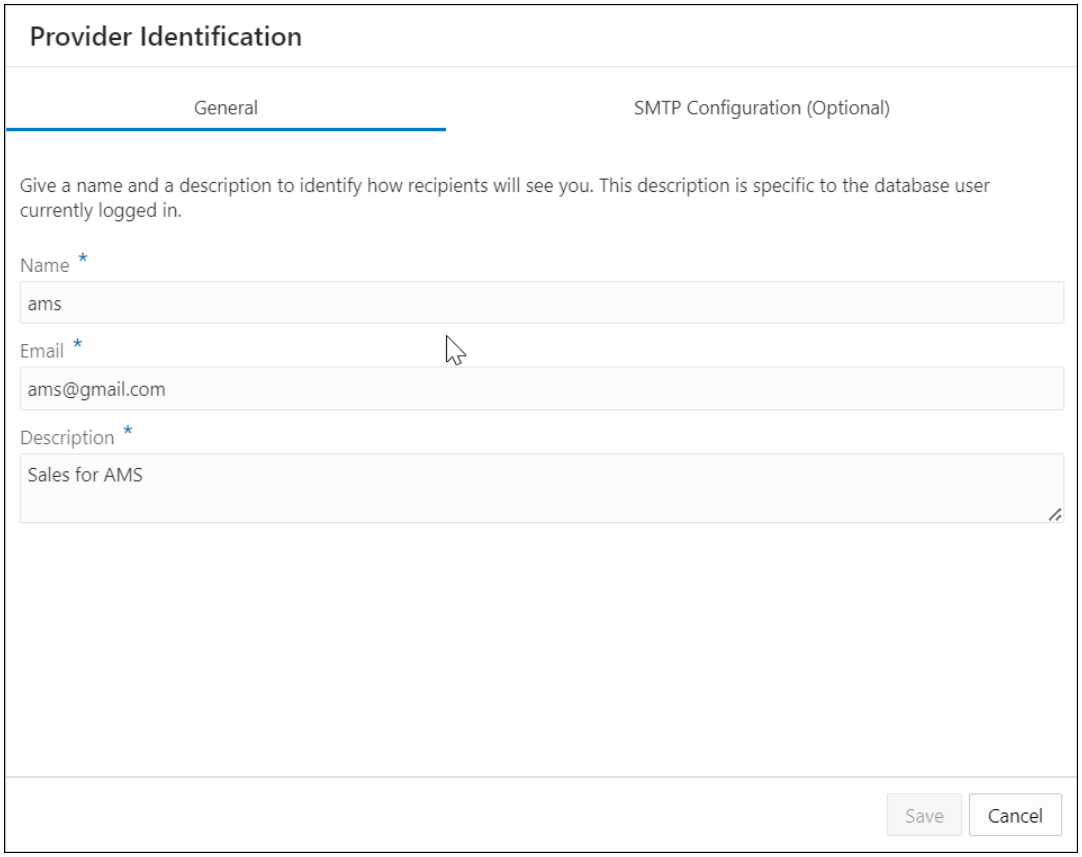
6. Log out and log in again to update privileges.
7. You now view the following **Provider** and **Consumer** page.



You have now enabled the share. Click **Provide Share**. On the Provide Share page, click **Provider Identification** to create a Provider ID. This provides information to the recipient on how to identify you.

This field provides the details of the provider before you share the data. The share provider identification will be available to recipients with whom you grant the share.

Selecting this button opens a Provider Identification wizard. Specify the following fields on the General tab of the Provider Identification wizard:



- **Name**: Enter the name of the provider. For example, *ams*.
- **Email**: Enter the email address of the provider. For example, *ams@gmail.com*.
- **Description**: Enter a description of the provider. For example, *AMS sharing stuff*.

Progress to the Simple Mail Transfer Protocol (SMTP) Configuration tab of the wizard. This tab configures email service like a local email client on your system (e.g., Thunderbird).

> **Note:**
>
> You must configure SMTP only once and then the system uses the saved configuration from that point forward.

Specify the following field values in the SMTP Configuration tab of the wizard:

- **Server host**: Enter the endpoint used to send the email. For example, *internal-mailrouter.oracle.com*.
- **Server port**: Enter the SMTP port used to accept an email. Email Delivery supports TLS on port 25. Sender: Enter the email address of the sender. For example, *oaapgens_us@oracle.com*.
- **Server Encryption**: This field indicates if TLS, the standard means of performing encryption in transit for emails, is being used. Providers must encrypt emails while it's in transit to the Oracle Cloud Infrastructure Email Delivery service. Encrypted emails are protected from being read during transit. If there is no encryption, enter, *None*.

Select credential to use for SMTP connection from the drop-down. If there is no available credential in the drop-down, you can create credential by clicking **Create Credential** icon. Refer to Create Oracle Cloud Infrastructure Native Credentials for more details.

Click **Test** to test the SMTP configuration. You will view a screen that asks you to run the ACL script. You can run the script if you have the ADMIN rights. This is just a first time setting. After you receive a successful message of SMTP Test, you can save the SMTP configuration.

You can now start sharing your data.

Provide Share

Use the Provide Share page to view the number of shares of the database, and the number of share recipients, search for a share entity, search for a recipient, create a new share, and create a new recipient.

The Provide Share page consists of widgets Shares and Recipients and icons to create Shares and Recipients. The page is described in the following sections:

- Provide Share Overview
  Use the Provide Share to create and publish shares.
- Provide Versioned Share
  In this type of share, data is shared and published as well-defined, known, as-of-snapshots. At the time of publication, the tool generates and stores the data share as parquet files in the specified bucket. The recipient can directly access the share in the object store.
- Provide Live Share
  In this type of share the recipient accesses data directly from the Oracle table or view. This share gives the recipient the latest data. Live Share works by querying data over a cloud link, which is implemented on a database link. The database link allows you to query objects over the network on a foreign machine and schema. The advantage of this mode is that the data is up to date as of the time of query.
- View Share Entity Details
  Use the **Actions** icon at the right of the share entity entry to view details about the live share or delta share entity you create.
- Create Share Recipient
  Use the **+ Create Share Recipient** icon on the Provide Share page to create a consumer of the data share. When there is already a recipient created, use the **+ Create Recipient** icon on the Recipients widget of the Provide Share page to create a consumer of the data share.
- View Share Recipient Entity Details
  Use the **Actions** icon at the right of the share recipient entity entry to view details about the live share or delta share recipient entity you create.
- Provide Share Overview
  The Provide Share page enables you to view, create, edit, and access information on the shares and recipients you create from this page.
- Provide Versioned Share
  You can share data as a set of distinct versions, such as at the end of each day. The recipient will only see changes to the data when you publish a new version. As a versioned share Provider, you must create an OCI native credential and associate the bucket's URL with the credential.

- Provide Live Share
  You can share data as of the latest database commit with Autonomous Databases in the same region. The recipient will always see the latest data.
- View Share Entity Details
  To view details about an entity, click the **Actions** icon at the right of the Share entity entry, then click **View Details**.
- Create Share Recipient
  On the Provide Shares page, click the **Recipients** widget and select **+ Create Recipient** to create a new Share Recipient.
- View Share Recipient Entity Details
  Click the **Actions** icon at the right of the Share Recipient entity entry, then click **View Details**.

Provide Share Overview

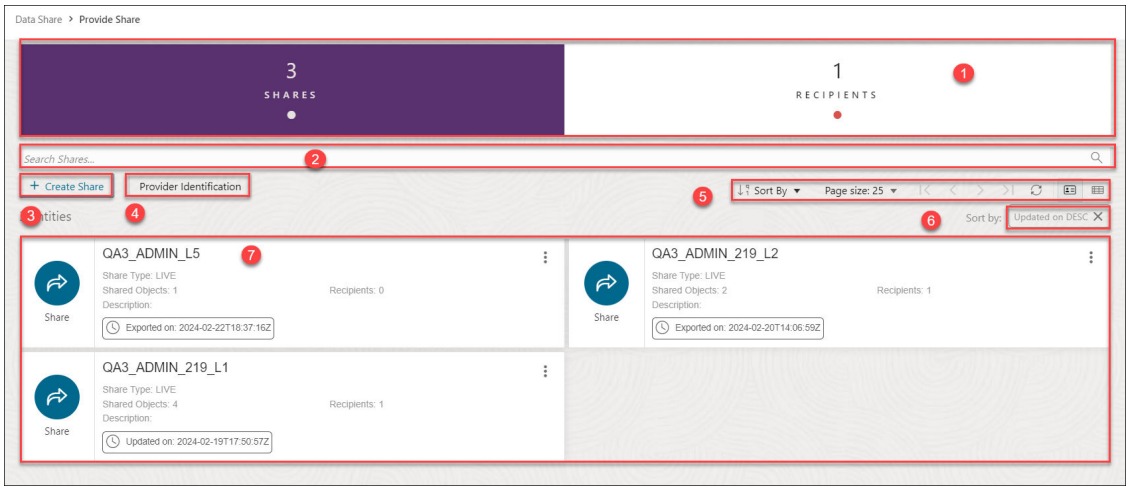
The Provide Share page enables you to view, create, edit, and access information on the shares and recipients you create from this page.

Click the **Provide Share** menu under Data Share in the Data Studio menu to reach the Provide Share page. You can switch the display of the Provide Share screen based on the widget you select from the Shares and Recipients section.

The search field and the display area of the **Provide Share** page varies when you switch between the **SHARES** and **RECIPIENTS** widgets.

Shares Page

The following image displays the Provide Share page when you select the **SHARES** widget.



The Provide Share page when you select the Shares widget consists of the following:

1. The top section of the Provide Share page displays Shares and Recipients section with two widgets. The widgets are defined in the following sections:
   - **SHARES**- Select this widget to view, search and perform actions on the Shares you create.
   - **RECIPIENTS**- Select this widget to view, search and perform actions on the Share Recipients you create.
2. **Search Shares** field

Selecting Shares widget enables you to search for the share you create. You can click the field and type or paste the name of the share or the recipient you are looking for. Click the magnifier icon to return the search input. Click **X** to clear your searches in the Search Shares field.

3. **+ Create Share** button
   You can use the **Create Share** wizard to define data shares based on object storage data and associate them with recipients. A single data share can consist of multiple shared objects that are shared with one or multiple recipients.

   You can create a new share and a new recipient from this button. Refer to the Provide Live Share, Provide Versioned Share and Create Share Recipients section to explore more on what each button does.

4. **Provider Identification** button : See Access and Enable the Data Share tool to view more about this icon for a first time user. If you have already configured SMTP, you can edit the Provider ID or set SMTP using this button.

5. **Toolbar**
   The toolbar consists of the following buttons:

   - **Sort By**
     To select sorting values, click the **Sort By** button to open the list of options. Then click the **Ascending** or **Descending** icon next to one or more of the sorting values.

     For example, if you select the **Ascending** icon next to **Entity name** and the **Descending** icon next to **Entity type**, the entities will be sorted in alphabetical order by entity name and then in reverse alphabetical order by entity type.

     Click **Reset** in the list to clear the choices in the list.

     The sorting values you choose are listed next to the **Sort by** label beneath the toolbar. Click the **X** icon on a sorting value to remove it.

   - **Page size**
     By default, up to 25 entities are displayed on the page. If you want more entities on a page, select a number from this list.

   - **Previous** and **Next**
     If the search results are displayed on multiple pages, click these buttons to navigate through the pages.

   - **Refresh**
     Click to refresh the entities shown on the page, based on the current search string.

   - **Entity view options**
     Choose one of these three options to set how entities are displayed on the page.

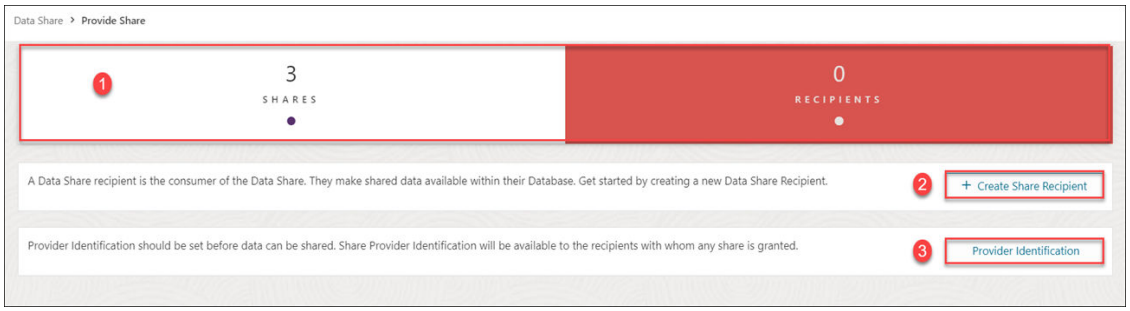
     Click **Open Card view** to display entities as card arranged into one or two columns; click **Open Grid View** to display entities as rows in a table; or click **Open List View** to display entities in a single column of panels.

6. **Sort by settings**
   When you set sorting values by using the Sort By control in the toolbar, the settings are displayed in small boxes beneath the toolbar. You can delete a setting by clicking the X icon in the box. Or you can change the settings by returning to the Sort By control in the toolbar.

7. **Display area**
   The area below the Search field displays the entities returned by a search. If you select the Shares widget, you will view the Share entities in the display area. If you select the Recipient widget, you will view the Share Recipient entities in the display area. This area shows all providers you are subscribed to.

Recipients Page

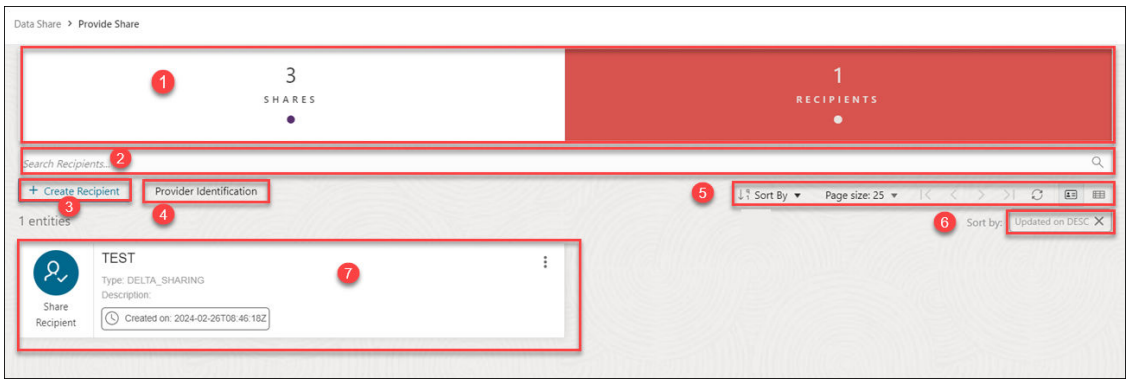
The following image displays the Provide Share page when you select the RECIPIENTS widget and ***there is no RECIPIENT created***



Click **+ Create Share Recipient** to create a share recipient. See Create Share Recipient for more details.

Click **Provider Identification** to specify the name and description to identify how the recipients will view you. You can also configure SMTP settings.

The following image displays the Provide Share page when you select the RECIPIENTS widget and ***there is minimum one RECIPIENT already created***



The Provide Share page consists of the following:

1. The top section of the Provide Share page displays Shares and Recipients section with two widgets. The widgets are defined in the following sections:
   - **SHARES**- Select this widget to view, search and perform actions on the Shares you create.
   - **RECIPIENTS**- Select this widget to view, search and perform actions on the Share Recipients you create.
2. **Search Recipients** field
   Searching for a share or recipient depends on the widget you select from the Shares and Recipients section. If you select the Recipients widget, you can search for the recipients you create. You can click the field and type or paste the name of the share or the recipient you are looking for. Click the magnifier icon to return the search input. Click **X** to clear your searches in the Search Shares or Search Recipients field.
3. **+ Create Recipient** button

You can create a new share and a new recipient from this button. Refer to the Provide Live Share, Provide Versioned Share and Create Share Recipients section to explore more on what each button does.

4. **Provider Identification** button: See Access and Enable the Data Share tool to view more about this icon for a first time user. If you have already configured SMTP, you can edit the Provider ID or set SMTP using this button.
5. **Toolbar**
   The toolbar consists of the following buttons:
   - **Sort By**
     To select sorting values, click the **Sort By** button to open the list of options. Then click the **Ascending** or **Descending** icon next to one or more of the sorting values.

     For example, if you select the **Ascending** icon next to **Entity name** and the **Descending** icon next to **Entity type**, the entities will be sorted in alphabetical order by entity name and then in reverse alphabetical order by entity type.

     Click **Reset** in the list to clear the choices in the list.

     The sorting values you choose are listed next to the **Sort by** label beneath the toolbar. Click the **X** icon on a sorting value to remove it.
   - **Page size**
     By default, up to 25 entities are displayed on the page. If you want more entities on a page, select a number from this list.
   - **Previous** and **Next**
     If the search results are displayed on multiple pages, click these buttons to navigate through the pages.
   - **Refresh**
     Click to refresh the entities shown on the page, based on the current search string.
   - **Entity view options**
     Choose one of these three options to set how entities are displayed on the page.

     Click **Open Card view** to display entities as card arranged into one or two columns; click **Open Grid View** to display entities as rows in a table; or click **Open List View** to display entities in a single column of panels.
6. **Sort by settings**
   When you set sorting values by using the Sort By control in the toolbar, the settings are displayed in small boxes beneath the toolbar. You can delete a setting by clicking the **X** icon in the box. Or you can change the settings by returning to the Sort By control in the toolbar.
7. **Display area**
   The area below the Search field displays the entities returned by a search. If you select the Shares widget, you will view the Share entities in the display area. If you select the Recipient widget, you will view the Share Recipient entities in the display area.

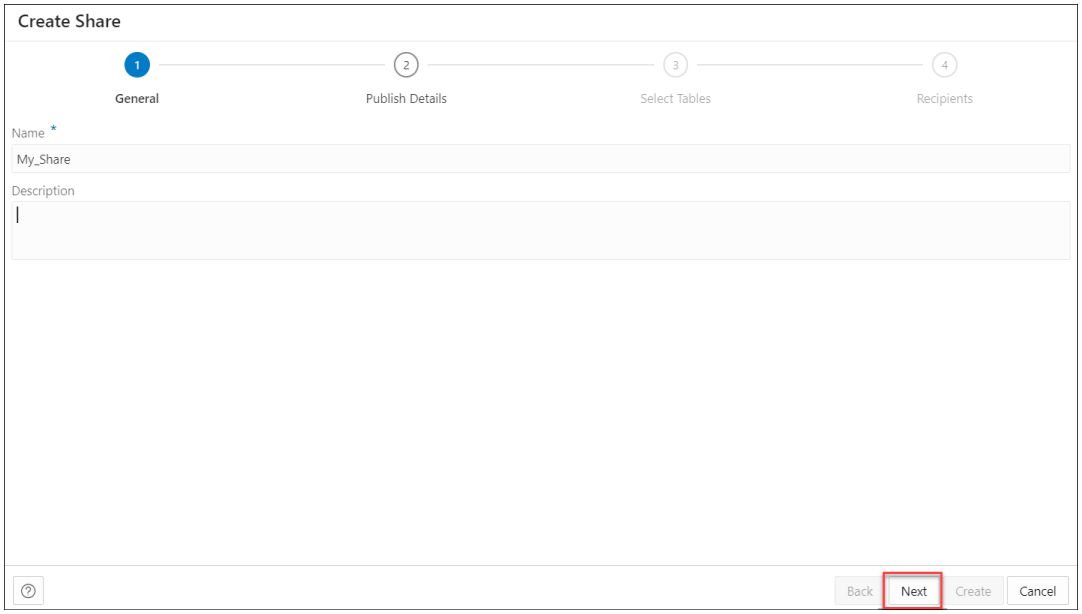
Provide Versioned Share

You can share data as a set of distinct versions, such as at the end of each day. The recipient will only see changes to the data when you publish a new version. As a versioned share Provider, you must create an OCI native credential and associate the bucket's URL with the credential.

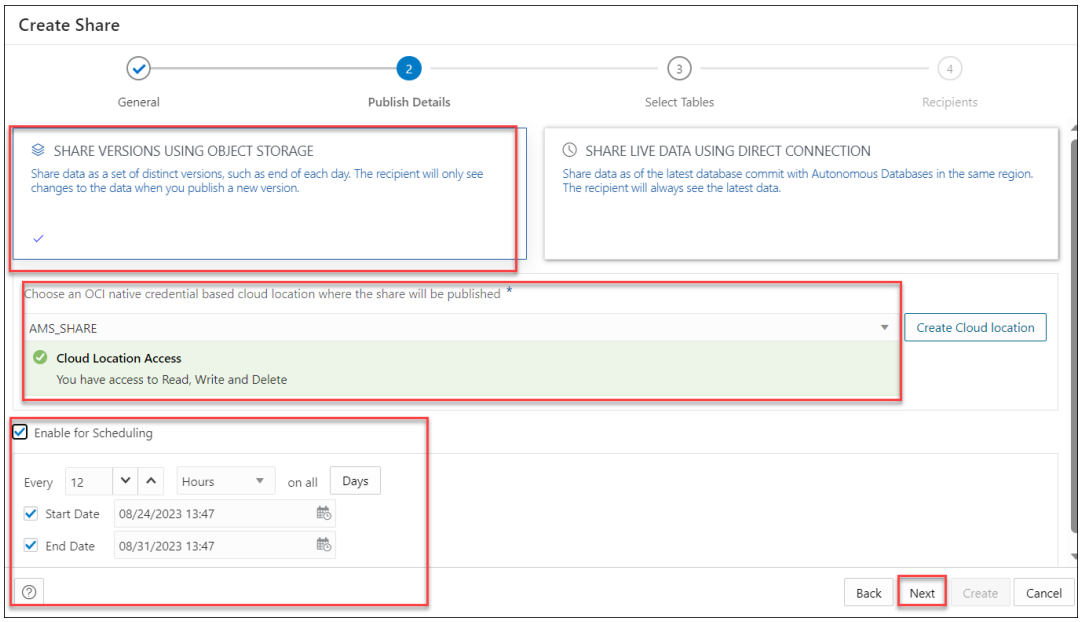
On the Provide Share page, select **+ Create Share** from the Create menu. This brings up the **Create Share** wizard.

1. On the Create Share wizard, in the **Name** field of the General tab, enter a name for the Share. For example: *My_Share*.
   In the **Description** field, enter a description for the link. For example: *Weekly Sales report*.

   Select **Next** to progress to the Publish Details tab of the Create Share wizard. You can alternatively click on the Publish Details tab to progress to the Publish Details screen of the wizard.



2. In the Publish Details Tables tab of the wizard, select **SHARE VERSIONS USING OBJECT STORAGE**.



   Select a cloud location where you want the share to be published. If you already have an OCI cloud storage location available, use the **Select Cloud location** drop-down and select any available Cloud location from the drop-down. You will view a successful message if you have access to read, write, or delete in that cloud location.

> **Note:**
>
> Select **Create Cloud location** and follow the steps described in the Create Oracle Cloud Infrastructure Native Credentials if you do not have a cloud location available.

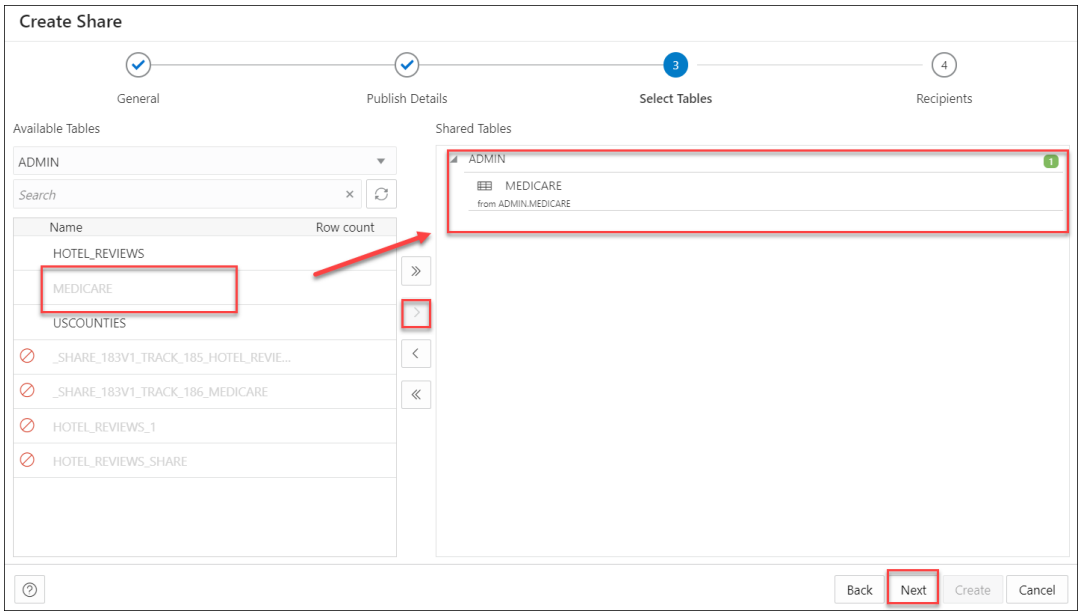
For sharing versioned data, you can enable the schedule for share. Select **Enable For Scheduling** option to set up a schedule for sharing. In the time interval fields, enter a number, and select a time type and the days on which to poll the bucket for new or changed files. For example, to poll every two hours on Monday, Wednesday, and Friday, enter **2**, select **Hours**, and then select **Monday**, **Wednesday**, **Friday** in the appropriate fields. You can select **All Days**, **Monday to Friday**, **Sunday to Thursday**, or **Custom** from the Week Days drop-down. The Custom field enables you to select Monday, Tuesday, Wednesday, Thursday and Friday in the appropriate fields.

Select a start and end date. If you don't select a start date, the current time and date are used as the start date. The end date is optional. However, without an end date, the share will continue to poll. This is an optional step.

Click **Next** to progress to the Select Tables tab of the Create Share wizard.

3. On the **Select Tables** tab of the wizard, select the schema from the drop-down menu and click the table you wish to share from the Available Tables. Choose any of the available options:

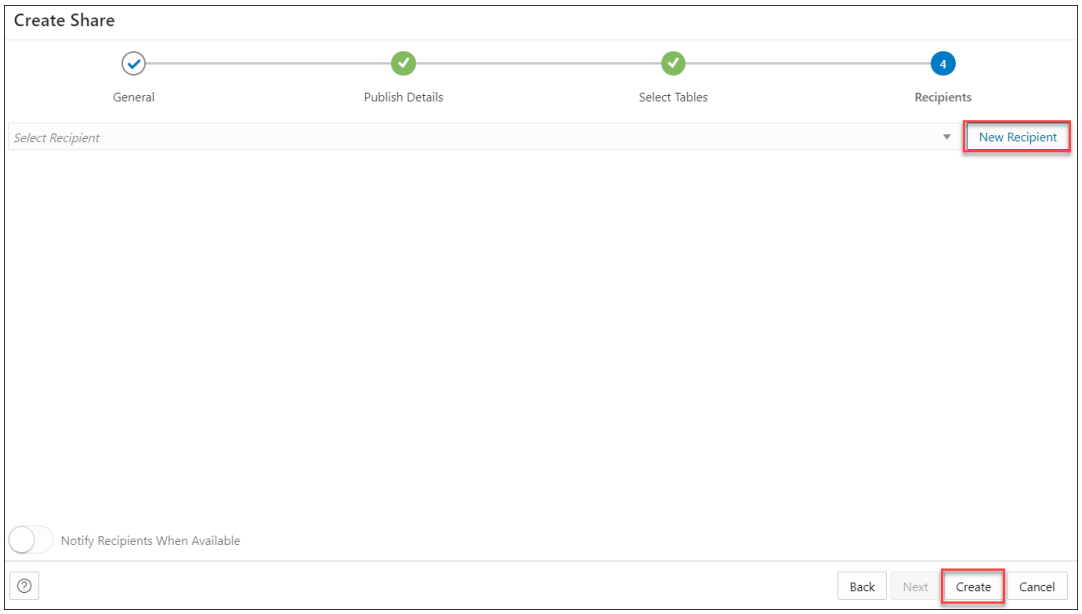
   - **>**: This option enables you to move the table to Shared Tables.
   - **<**: To remove the selected table from Shared Tables select this option.
   - **>>**: This option allows you to move all the tables to the Shared Tables screen.
   - **<<**: To remove all the selected tables from Shared Tables select this option.



Click **Next** to proceed to the Recipients tab of the Create Share wizard.

> **Note:**
>
> You may find it convenient to click on the specific tab to move back and forth. You can update changes on any screen this way.

4. On the Recipients tab of the Create Share wizard, select available Recipient from the drop-down.



Select **Notify Recipients When Available** when you want to notify the recipients when they are available. This can be enabled when SMTP is configured to send an email notification to the recipient.

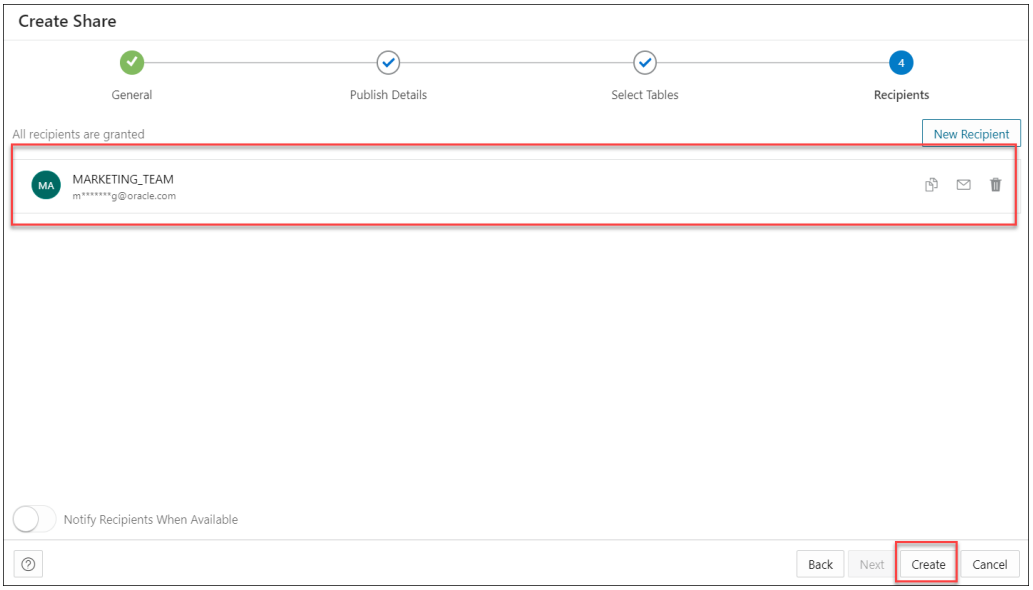
5. In case you want to create a new recipient, select **New Recipient**. Selecting New Recipient opens a **Create Share Recipient** wizard. See Create Share Recipient.

> **Note:**
>
> Ensure there are no spaces in the name of the recipient. This field does not support duplicate recipient names. If you try creating a recipient with a duplicate name, the creation of the share fails with the "Name is already used by an existing object" error.

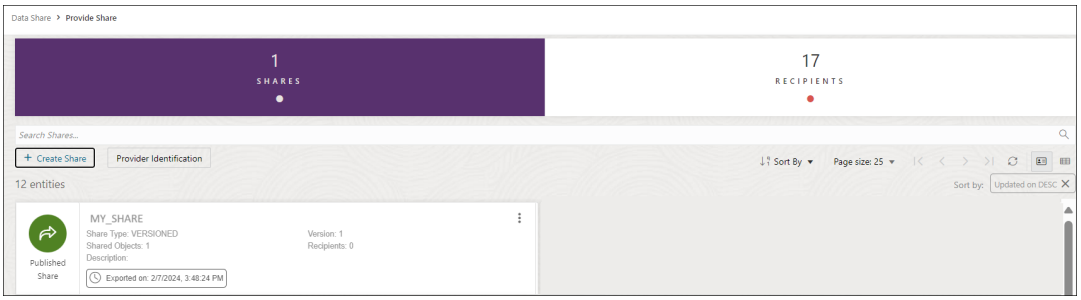
Click **Create** to finish creating the share recipient. The newly added recipient is displayed in the list of recipients. Click **Cancel** to cancel the ongoing process of recipient creation.

6. After you create a new share recipient, in the granted recipient's list you can:
   - Select **copy profile activation link to clipboard** icon to copy the activation link.
   - Select **Email recipient profile download link** icon to mail the profile link to the recipient you select in this step. Once you select it, the Share tool triggers an activation mail with the profile link to the recipient's email address. Click **Send** to share the profile link with the recipient. See *About the Consume Share Page* to know more about the activation mail and how to proceed as a recipient once you receive the activation mail.

- Click **deselect Recipient** icon to remove the selected recipient.



Select **Create** to create the share. Note that the status of the share changes from unpublished to published depending on the size of the table in the share. You will receive a successful share notification message at the top right of the display that says the share is successful. The Provide Share page displays the share along with its details such as the entity type, owner, shared objects and the recipients.



Note:

A versioned share will show different colors and states depending upon what state the share is in.

Provide Live Share

You can share data as of the latest database commit with Autonomous Databases in the same region. The recipient will always see the latest data.

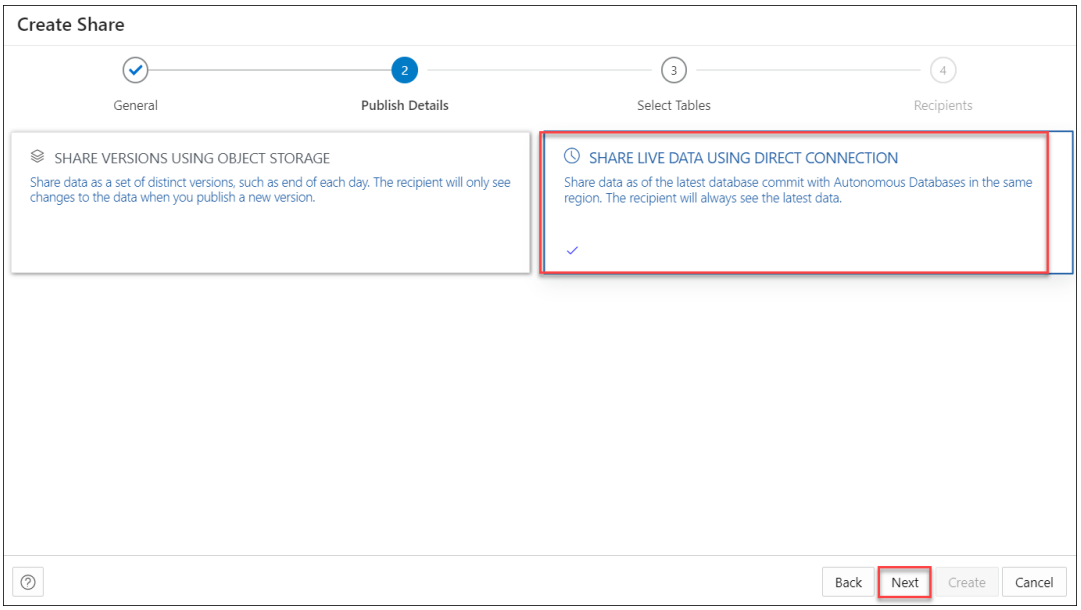
To provide access to live shared data, you must define data shares and grant recipients access to consume them. Use the **Create Share** wizard to create and publish shares for live data.

On the Provide Share page, select **+ Create Share** from the Create menu. This brings up the Create Share wizard.

1. On the Create Share wizard, in the **Name** field of the General tab, enter a name for the Share. For example: *My_Share*.
   In the **Description** field, enter a description for the data you are sharing. For example: *Weekly Sales report*.

   Select **Next** to progress to the Publish Details tab of the Create Share wizard. You can alternatively click on the Publish Details tab to progress to the Publish Details screen of the wizard.

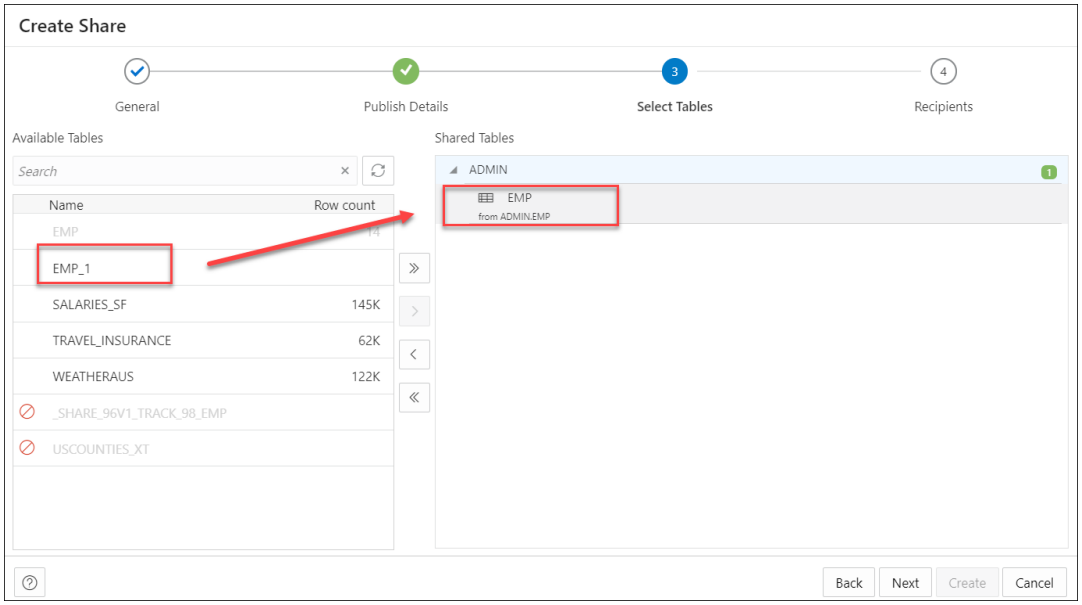
2. In the Publish Details Tables tab of the wizard, select **SHARE LIVE DATA USING DIRECT CONNECTION**.



   Click **Next** to progress to the Select Tables tab of the Create Share wizard.

3. On the **Select Tables** tab of the wizard, select the schema from the drop-down menu and click the table you wish to share from the Available Tables. Choose any of the available options:

   - **>**: This option enables you to move the table to Shared Tables.
   - **<**: To remove the selected table from Shared Tables select this option.
   - **>>**: This option allows you to move all the tables to the Shared Tables screen.
   - **<<**: To remove all the selected tables from Shared Tables select this option.

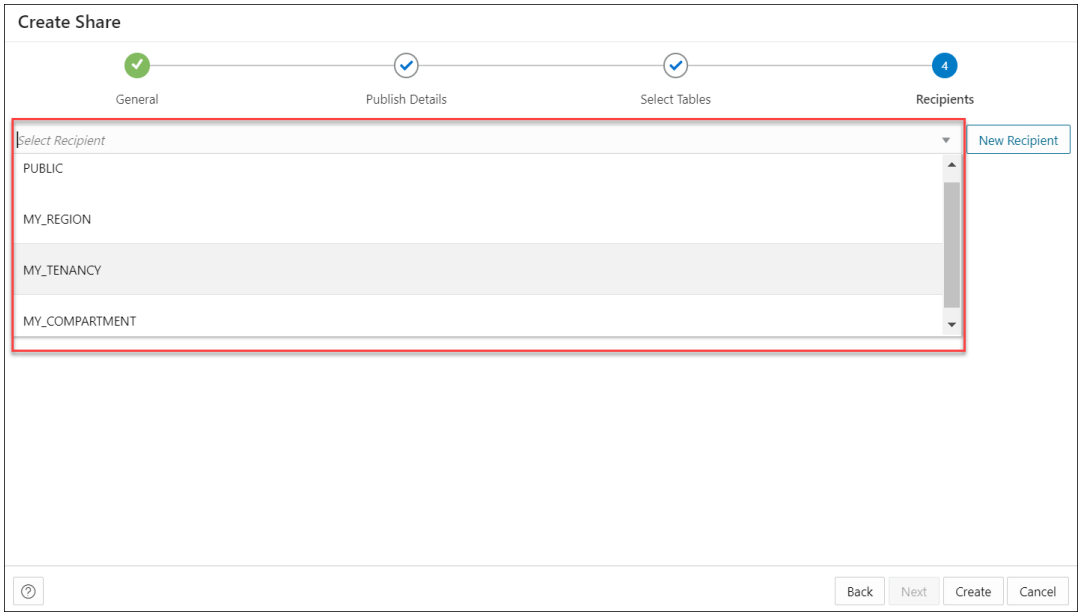


> **Note:**
>
> If a live share data provider is sharing multiple tables which would normally be joined together, it is recommended that the producer create a view that performs the joins and then only shares the view.

Click **Next** to proceed to the Recipients tab of the Create Share wizard.

> **Note:**
>
> You may find it convenient to click on the specific tab to move back and forth. You can update changes on any screen this way.

4. On the Recipients tab of the Create Share wizard, select available Recipients from the drop-down. The list of recipients available in the drop-down depends on the scope of the share. You can select any or all the following values depending on with whom you intend to share the data:
   - **MY_REGION**: Access to the data share is allowed for databases in the same region as the data share provider.
   - **MY_TENANCY**: Access to the data share is allowed for databases in the same tenancy as the data share provider.
   - **MY_COMPARTMENT**: Access to the data share is allowed for databases in the same compartment as the data share provider.

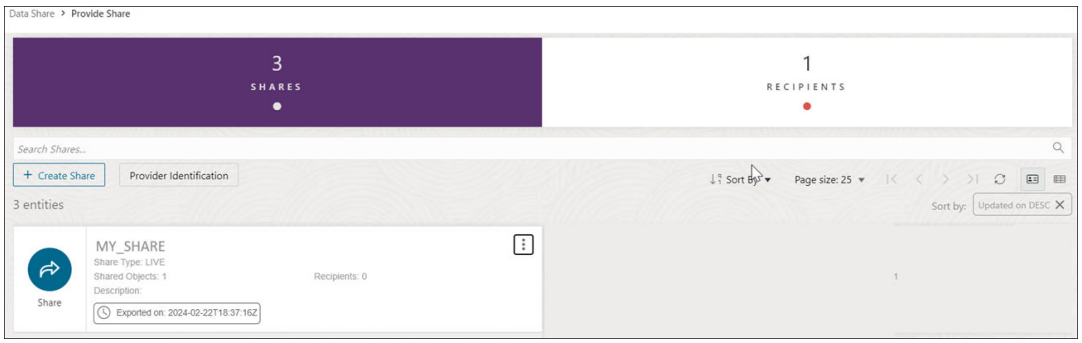


5. In case you want to create a new recipient, select **New Recipient**. Selecting New Recipient opens a **Create Share Recipient** wizard. Enter the recipient's Name, Description (optional), Sharing ID, Email in their respective fields.

> **Note:**
>
> - Ensure there are no spaces in the name of the recipient. This field does not support duplicate recipient names. If you try creating a recipient with a duplicate name, the creation of the share fails with the "Name is already used by an existing object" error.
> - A Sharing ID is a unique provider for your Autonomous Database. The Data Share tool uses it to share data with you. Copy the Sharing ID from the Consume Share page to the clipboard and paste it in the Sharing ID field of the **Create Share Recipient** wizard.

Click **Create** to finish creating the share recipient. The newly added recipient is displayed in the list of recipients. Click **Cancel** to cancel the ongoing process of recipient creation.

6. Select **Create** to create the share. You will receive a successful share notification message at the top right of the display that says the share is successful. The Provide Share page displays the share along with its details such as the entity type, owner, shared objects and the recipients.



Click the name of the entity or click **Action** to view details about the share entity. See View Share Entity Details.

After share is created, you can view the new Share entity in the Provide Share page.

View Share Entity Details

To view details about an entity, click the **Actions** icon at the right of the Share entity entry, then click **View Details**.

For all entities, the details include General, Selected Tables, Publishing details, Recipients, Versions, Log Details, Lineage and Impact sections.

General

The General tab displays the Name and description of the share.

Selected Tables

The Selected Tables tab displays the list of tables you wish to share.

Publishing

This tab displays the publishing details you configure during creation of the share. It displays whether you select live data share or share versioned data.

Recipients

You can review the list of recipients with whom you intend to share objects. It displays the name of the recipient with their mail address.

Jobs

See the Jobs option below to view information about it.

Log Details

You can view the detailed status of the share in this tab. It provides information such as, the time, Status, Log Level, and Details of each activity that takes place during the running of the job, i.e., creation of the share. It also displays if any errors are encountered during the creation of the share.

Lineage

Lineage displays all known information about the dependencies of the entity, and therefore how the entity was created and how it is linked to other entities.

For example, for a share you create in the database, the lineage is just the cloud location. For a share that you create by adding files from cloud storage, the lineage includes the cloud location of the file you share.

Pointing to the name of an item in the lineage displays the share name, the application that creates it, the type of entity, the path to it, and the schema it is in.

Arrows point from an entity to the entity that it derives from. For example, for a share you create, an arrow points from the cloud location to the share. If you point to an arrow, then a Links Information box appears that shows information about the relationship between the two entities.

To view more details about an item, click the **Actions** icon for the item, then click **Expand**. For a cloud location, the files in the location are displayed. Pointing to the name of the file displays the name, application, type, path, and schema of the file. To collapse the display, click the Actions icon, then click Collapse.

Impact

Impact shows all known information about the downstream use of an entity, and therefore how a change in the definition of an entity may affect other entities that depend on it.

For a specific share entity, you can perform the following actions using the Actions context menu:

- **View Details:** See View Share Entity Details.
- **Objects:** Opens Select Tables dialog of the specific share where you can remove tables from the list of Shared Tables list or add tables from the list of Available Tables list to the list of Shared Tables.
- **Recipients and Profiles:** Opens the Recipients and Profiles dialog which enables you to create a new share recipient. You can also revoke the recipient right from the share by clicking X next to the recipient name. The Recipients and Profiles dialog enables you to update the list of recipients, copy profile activation link, send an activation mail to the selected recipient.
- **Jobs:** Opens the Jobs table and displays information related to it. A job is created whenever something needs to happen to a share. This includes the following:
  - PUBLISH VERSIONED SHARE
  - PUBLISH LIVE SHARE
  - DROP SHARE VERSIONS

  Only one job can run at a time on a single share. E.g. if the user calls `publish_share` while a `PUBLISH` job is already running, then the second job will have to wait. The `STATUS` column reflects the same.

  The `JOB TYPE` column displays the method of parquet files generation (only for a versioned share). It's values can be `ODI` or `DBMS_CLOUD`.

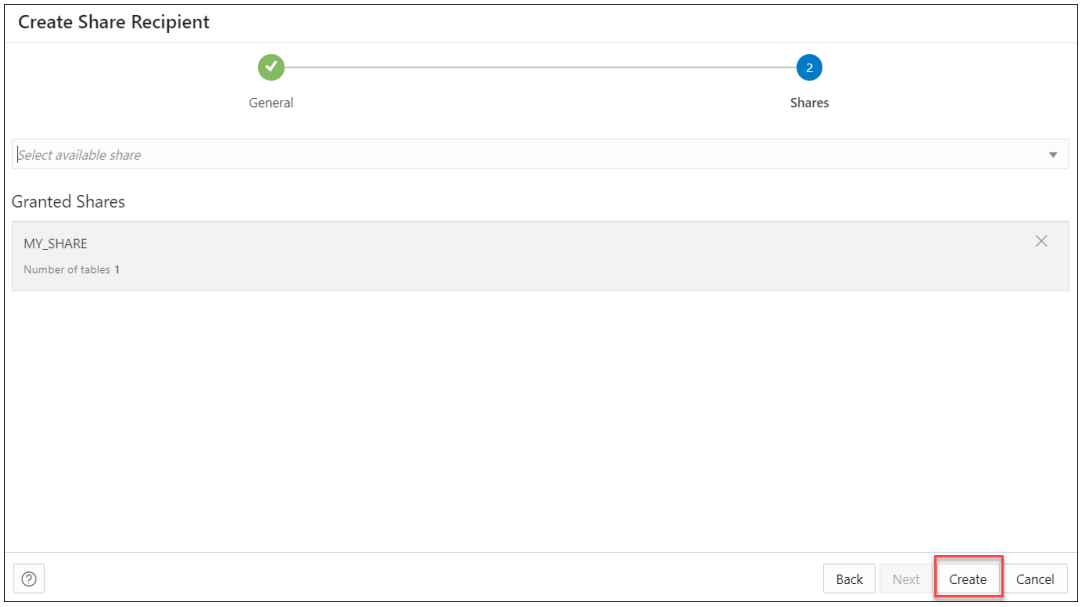
  For more details, refer to the USER_SHARE_JOBS View.
- **Publish:** Select **Publish** to publish the updated share.
- **Unpublish:** Select **Unpublish** to unpublish the share. This option prevents all recipients from accessing a single share, but lets them continue to access any other shares. You can restore access by publishing share again.
- **Delete:** Deletes the Share Provider Entity.

Create Share Recipient

On the Provide Shares page, click the **Recipients** widget and select **+ Create Recipient** to create a new Share Recipient.

Share Recipients are those with whom we intend to share objects with. You can share objects securely with users outside of Oracle workspace. Apart from creating recipients from the Create Share wizard, you can also create a share recipient from the Create Recipient icon on the Provide Share page and verify the details you configure while its creation.

1. On clicking the **+ Create Recipient** icon, a Create Share Recipient dialog box appears.



2. On the General tab, specify the following fields:

   - **Name**: Enter the name of the Share Recipient. For example, *Marketing_team.*

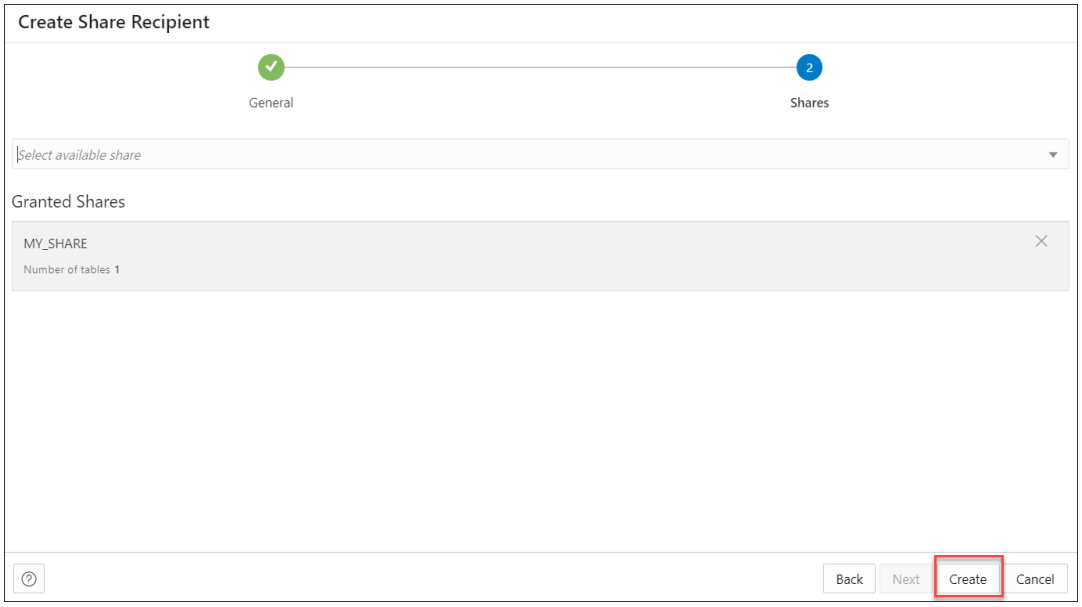
   > **Note:**
   >
   > Ensure there are no spaces in the name of the recipient. This field does not support duplicate recipient names. If you try creating a recipient with a duplicate name, the creation of the share fails with the "Name is already used by an existing object" error.

   - **Email**: Enter the email ID of the recipient. For example, *marketing@oracle.com*.
   - **Description**: Add a description. This is optional.
   - **Sharing ID**: An intended Recipient should copy this ID from the consumer page and send it to the Producer to be used here. The Share Provider wont have access to the consumer page. **This field is required for Live Share Recipients only**.
     On the User token lifetime section, enter any value in the `TOKEN_LIFETIME` field for the share recipient. This property specifies for how long the generated token will be valid after which the recipient loses access to the data share and must request a new token. Enter 60 in the first text field. Select Minutes from the drop-down menu.

   Click **Next** to progress to the Shares tab of the Create Share Recipient dialog box.

3. On the Shares tab, select an available share from the drop-down. As you select the share, it lists under the Granted Shares section. You have the choice to share multiple shares to

the recipient.



Select **Create** to create the Share recipient.

Once you have created the Share recipient, you will be able to view the newly created Share Recipient entity in the display area of the Provide Share page. The new recipient card displays details such as the entity type and the owner.

Click the name of the entity or click **Action** to view details about the share recipient entity. See View Share Recipient Entity Details.

View Share Recipient Entity Details

Click the **Actions** icon at the right of the Share Recipient entity entry, then click **View Details**.

For all entities, the details include general details, Log details, Granted Shares, Lineage and Impact.

General

The General tab displays the Name and Email address of the Share Recipient.

Granted Shares

The Granted Shares tab displays the list of shares granted to the recipient.

Log Details

The Log details tab displays the Time, Status, Log Level, Share Name, Request, End Point, Remote address, User Agent and Details of the Recipient.

Lineage

Lineage displays all known information about the dependencies of the entity, and therefore how the entity was created and how it is linked to other entities.

For example, for a share you publish, the lineage includes the files in cloud storage along with the cloud location of the files you share.

To view more details about an item, click the **Actions** icon for the item, then click **Expand**. For a cloud location, the files in the location are displayed. Pointing to the name of the file displays the name, application, type, path, and schema of the file. To collapse the display, click the **Actions** icon, then click **Collapse**.

Impact

Impact shows all known information about the downstream use of an entity, and therefore how a change in the definition of an entity may affect other entities that depend on it.

Click **Actions** in the Share Recipient entity card to open the context menu. The actions available are:

- **View Details:** See View Share Recipient Entity Details.
- **Granted Share:** Opens Manage Granted Shares dialog where you can grant access to a share from list of available shares or you can revoke access from the current selection of share.
- **Get Bearer Token:** Opens an Auth Token dialog to authenticate the recipient. Use this option when your token expires. This enables you to generate a new token. A Bearer Token is part of the JSON file.
- **Send Activation Mail:** Opens the mail with the profile link.
- **Copy Profile Activation Link to Clipboard:** Copies the JSON profile activation link to clipboard.
- **Rename:** Select this option to rename the Recipient.
- **Delete:** Select this option to delete the recipient you created entirely. By doing so, access to all shares will end instantly and any existing delta credentials will become invalid.

Consume Share

Once the providers share the objects, there are a few steps the recipients need to follow to consume the share.

Use the Consume Share page to perform the following operations:

- Consume Share Overview:
  To consume data shares, you need to subscribe to them and create views of tables included in the live share.
- Consume Versioned Share:
  As a recipient, you will need to download your share profile, subscribe to the data share provider, register the shares and create external tables on top of your shares. The Data Share tool authorizes the access using the JSON profile sent to the recipient using the activation mail. After the access is granted, the Data Share tool links the shared objects with the Data link tool where the consumer can run the data link job and access the objects shared by the provider.
- Consume Live Share:

This lets you as a recipient to consume live data from the database.

- View Share Provider Entity Details:
  Use the **Actions** icon at the right of the live share or delta share provider entity entry to view details about the live share or delta share provider entity you create.
- Consume Share Overview
  The Consume Share provides an overview on the list of share providers, search for share providers and add a share provider.
- Consume Versioned Share
  You must follow these steps to make shared versioned data available to you within Oracle Autonomous Database. Data shared with you through Delta Sharing is not automatically available and discoverable in your Autonomous Database.
- Consume Live Share
  Live data shared with you through data sharing is not automatically available for consumption.
- View Share Provider Entity Details
  To view details about the Share Provider entity, click the **Actions** icon at the right of the Share Provider entity entry, then click **View Details**.

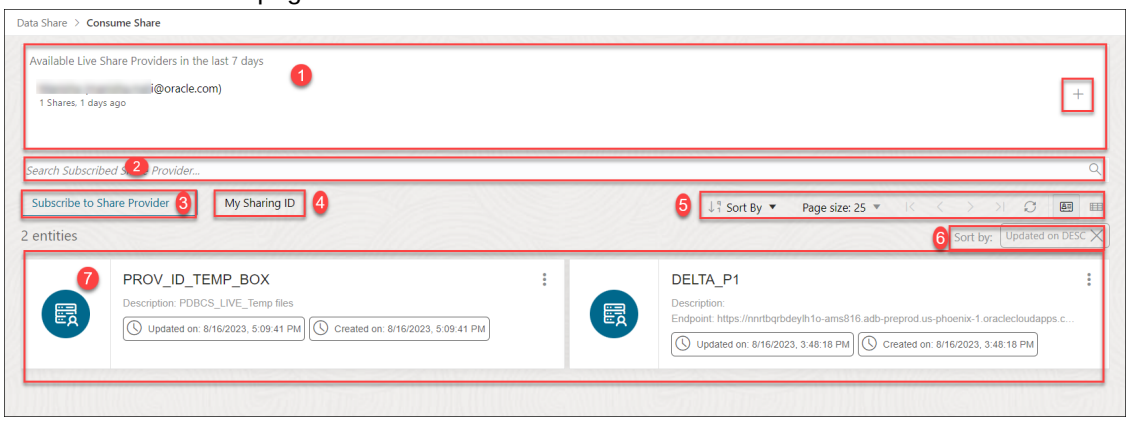
Consume Share Overview

The Consume Share provides an overview on the list of share providers, search for share providers and add a share provider.

To navigate to the Consume Share page, do either of the following:

- On the Data Studio menu, select **Consume Share** under the Data Share menu.
- On the Data Share page, click the **Consume Share** widget present in the Provider and Consumer section.

The Consume Share page contains:



1. **Available Live Share Providers in the last 7 Days**
   This area displays the list of available Live Share Providers you create in the last 7 days. You can update any of the fields as per your wish. You can also subscribe to Live Share using the + sign to the right of Shares listed in **Available shares in the last 7 days**.
2. **Search Subscribed Share Provider** field

   You can search for the Share Recipient you create by entering the name of the Subscribed Share Provider. Enter the name of the Subscribed Share Provider, for example, *REVIEW_PROVIDER* and click the **magnifer** icon to complete the search. The Share tool displays the search results in the display area.

3. Select **+ Subscribe to Share Provider** to subscribe to a new Share Provider. See Subscribe to Share Provider for exploring this icon.
4. **My Sharing ID**
   A Sharing ID is a unique provider for your Autonomous Database. Copy this ID to the clipboard and paste it in the Sharing ID field of the **Create Share Recipient** wizard. This enables a Live Share to be shared with a share provider.
5. **Toolbar**
   The toolbar consists of the sort by, page size, refresh and entity view options.
6. **Sort by settings**
   When you set sorting values by using the Sort By control in the toolbar, the settings are displayed in small boxes beneath the toolbar. You can delete a setting by clicking the **X** icon in the box. Or you can change the settings by returning to the Sort By control in the toolbar.
7. **Display area**

   The area beneath the Search Consumer Share Providers field displays the entities returned by a search and that match the filter criteria set in the Filters panel. You can sort the entities by clicking the **Sort By** button and then setting sort values.

Consume Versioned Share

You must follow these steps to make shared versioned data available to you within Oracle Autonomous Database. Data shared with you through Delta Sharing is not automatically available and discoverable in your Autonomous Database.

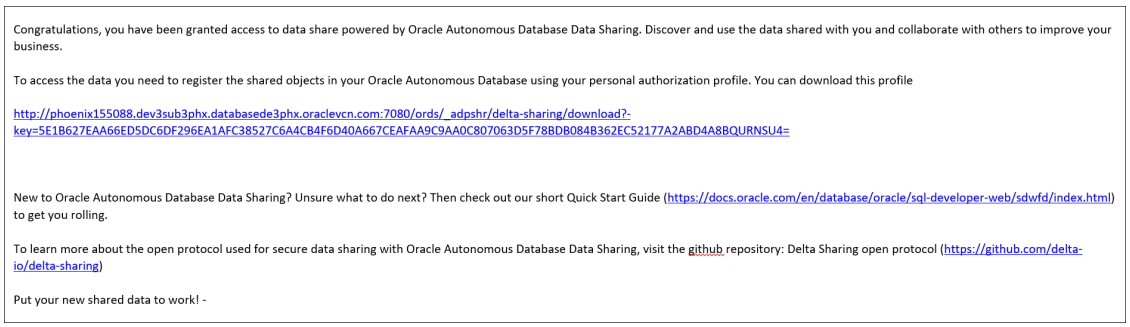
You need to perform several basic steps to subscribe and access the provided data shares. Here are the steps you need to perform:

- Download the JSON profile.
- Subscribe to the data share provider.
- Register shares made available to you.
- Create external tables on top of your shares.

About JSON Profile

Profile files are JSON files containing a user's credentials to access a Delta Sharing Server. This enables you to authenticate yourself with the delta sharing server and to discover the data shares you are eligible to access. Download the profile using the URL included in the invitation email sent by the data share provider.

Once the Share tool creates a share object, the recipient receives an activation mail with subject titled *Oracle Autonomous Database Data Share*.



Congratulations, you have been granted access to data share powered by Oracle Autonomous Database Data Sharing. Discover and use the data shared with you and collaborate with others to improve your business.

To access the data you need to register the shared objects in your Oracle Autonomous Database using your personal authorization profile. You can download this profile

http://phoenix155088.dev3sub3phx.databasede3phx.oraclevcn.com:7080/ords/_adpshr/delta-sharing/download?-key=5E1B627EAA66ED5DC6DF296EA1AFC38527C6A4CB4F6D40A667CEAFAA9C9AA0C807063D5F78BDB084B362EC52177A2ABD4A8BQURNSU4=

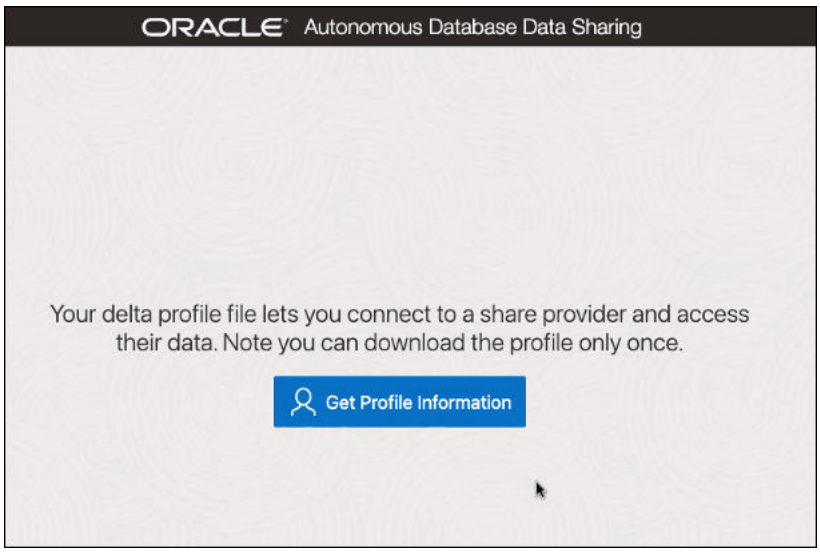
New to Oracle Autonomous Database Data Sharing? Unsure what to do next? Then check out our short Quick Start Guide (https://docs.oracle.com/en/database/oracle/sql-developer-web/sdwfd/index.html) to get you rolling.

To learn more about the open protocol used for secure data sharing with Oracle Autonomous Database Data Sharing, visit the github repository: Delta Sharing open protocol (https://github.com/delta-io/delta-sharing)

Put your new shared data to work! -

To access the share, you need to register the shared objects using personal authorization JSON profile.

You can click on the profile link to download the JSON profile. Clicking the profile link takes you to a new screen in the browser with a **Get Profile Information** button as shown below:



Select **Get Profile Information** to download the JSON profile to connect to the share provider.

> **Note:**
>
> You can click on the **Get Profile Information** button only once. The share tool does not allow you to select **Get Profile Information** twice. Clicking on it twice brings up a screen which displays the list of causes of the failure to download the profile.

The below profile is an example of the JSON profile you download.

```
{  "
        shareCredentialsVersion ": 1,
        "endpoint": "https://myhost.us.example.com/ords/prov/
_delta_sharing/",
        "tokenEndpoint": "http://myhost.us.example.com:1234/ords/pdbdba/oauth/
token",
        "bearerToken": "-xxxxxxxxxxxxxxxxxxxx",
        "expirationTime": "2023-01-13T07:53:11.073Z",
        "clientID": "xxxxxxxxxxxxxxxxxxxxx..",
        "clientSecret": "xxxxxxxxxxxxxxxxxxx.."}
```

The profile stores the credentials in an encrypted format. The parameters with their description are:

- **shareCredentialsVersion:** The version of the share you publish.
- **endpoint:** Specifies the share endpoint.
- **tokenEndpoint:** Specifies the token endpoint. The Share tool client use the token endpoint to refresh the timeout on your bearer token if you are consuming the share using Oracle.

- **bearerToken:** This is a cryptic string which the authentication server generates in response to a login request.
- **expirationTime:** This is the time taken for the authentication to expire.
- **ClientID:** Specifies the public identifier the authentication server generates when you register the instance for authentication.
- **clientSecret:** Specifies a secret identifier the authentication server generates for authorization.

Copy the JSON content of the profile in a notepad. You will need this JSON below to subscribe your share provider.

> **Note:**
>
> Ensure you copy the full JSON content profile, including the left brace and the right brace.

Security enhancements

As a share recipient, you must set up an access control list (ACL) to the share provider's machine by using the `APPEND_HOST_ACE` procedure as an ADMIN user, or another privileged user. This allows you to access the share via the Internet.

> **Note:**
>
> This must be done before using the Add Share Provider Wizard to add an access control entry (ACE) to the access control list (ACL) of the host (i.e. Share provider). You can find the host name from the JSON profile you downloaded in the previous step.

For example, if you wish to allow a database user, A_SHARE_USER to access the endpoints on a host (Share provider) named, here is a sample of PL/SQL procedure you will need to run in the SQL worksheet editor as an admin. As a prerequisite, extract the host name from the endpoint property in the delta sharing JSON profile, as provided in the example above. The hostname from the example is `myhost.us.example.com`.

```
BEGIN
        dbms_network_acl_admin.append_host_ace(
        host =>'myhost.us.example.com',
        lower_port=>443,
        upper_port=>443,
        ace => xs$ace_type(
        privilege_list => xs$name_list('http', 'http_proxy'),
        principal_name =>'A_SHARE_USER',
        principal_type => xs_acl.ptype_db));
        COMMIT;
        END;
        /
```

Following are the parameters with their description:

- **host**- Specifies the name or the IP address of the host. The host or domain name is not case-sensitive.
- **lower port**- Specifies the lower port of an optional TCP port range.
- **upper port**- Specifies the upper port of an optional TCP port range.
- **ace** – The Access Control Entry.
- **privilege list**- Specifies the list of network privileges to be granted or denied.
- **principal_name**- It is the principal (database user or role) to whom the privilege is granted or denied. It is case-sensitive.
- **principal_type**- Specifies the type of principal you use.

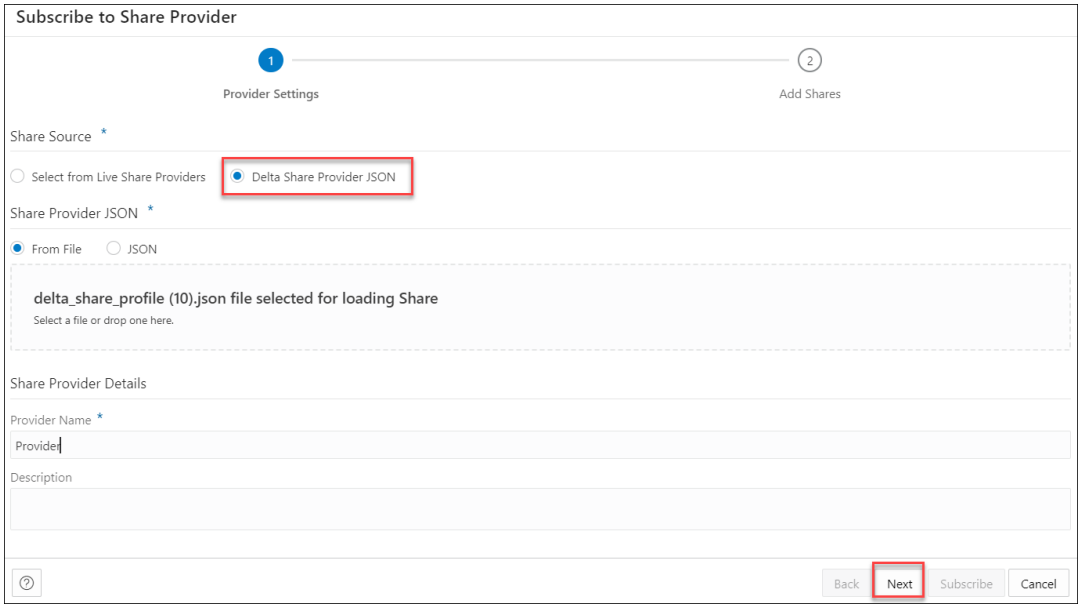
Refer to the PL/SQL Packages and Types Reference document for more details on the `DBMS_NETWORK_ACL_ADMIN` package subprograms.

Grant an ACL to the user on the local ORDS endpoint. You will need this to generate bearer tokens on locally created shares.

```
PRIV_ORDS_ACL              CONSTANT PLS_INTEGER := 8;
```

In this process you will load the provider's JSON profile for configuration and credentials to enable access to the recipients.

1. Open the Consume Share page and click **+ Subscribe to Share Provider** to select **Subscribe to Delta Share Provider** from the drop-down. This opens the Subscribe to Share Provider dialog box.
2. On the Provider Settings pane of the Register Share Provider dialog box, specify the following details:



   - **Provider Name**: *Provider*.
   - **Description**: Add a description. This field is optional.

   Under Share Source section, choose **Delta Share Provider JSON**.

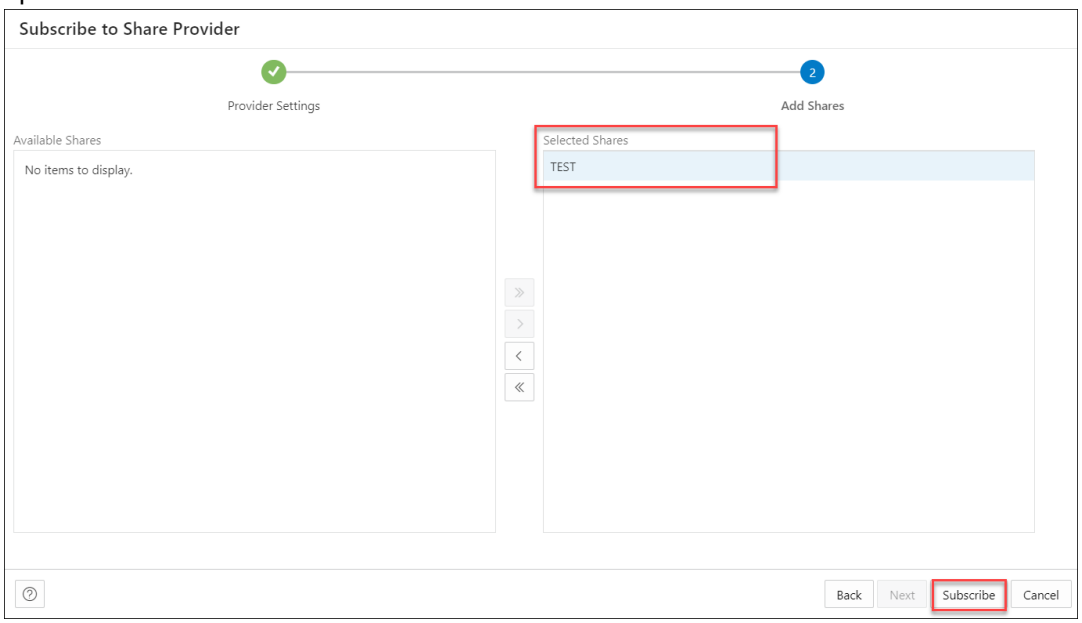
   Under Share Provider JSON field, you can share the JSON profile in the following ways:

- **From File**: Select this option and click on the drop area titled "Delta Share Profile JSON". Clicking on the area opens your local repository where you can select the JSON profile you had downloaded.
- **JSON**: You can select this option and paste the JSON content of the profile you copy into the notepad.

Upload the JSON profile file and create a share provider subscription.

Click **Next** to progress to the Add Shares tab.

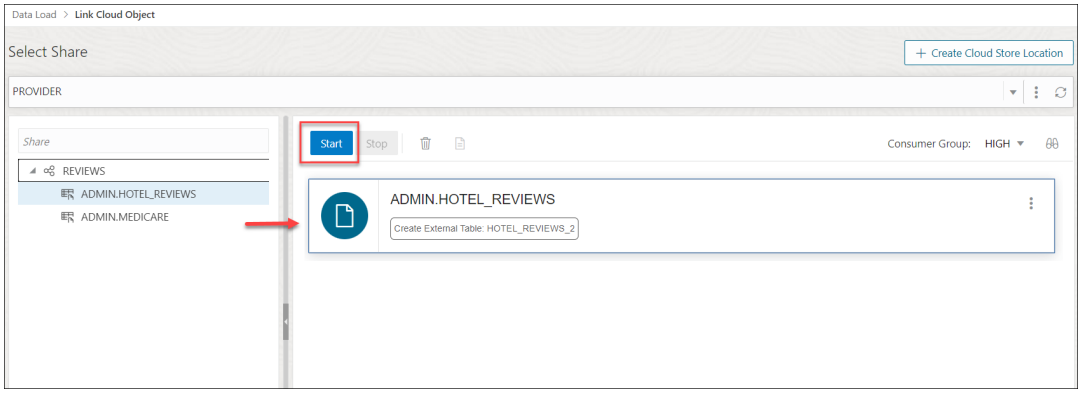
3. On the Add Shares tab of the dialog, you will view the list of available shares. Click the share you wish to consume from the Available Shares and select any of the available options:



- **>**: This option enables you to move the Available Share to Selected Shares.
- **<**: Select this option to remove the selected share from Selected Shares.
- **>>**: This option allows you to move all the shares to the Selected Shares screen.
- **<<**: Select this option to remove all the selected shares from Selected Shares.

4. Click **Subscribe** to add the share. A confirmation prompt appears when the provider is created successfully. After successful creation of the provider, you will now view the Link Cloud Object screen of the Data Load page.

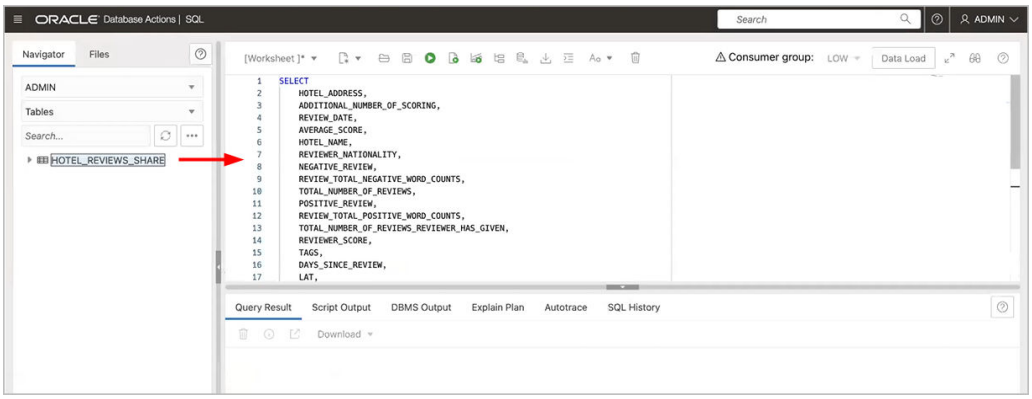
5. You can view the name of the Share provider in the cloud storage location field. The share appears in the source file location with the files you add to the share.



Expand the Share folder cart, drag and drop the file you share from the source to the Data Link cart.

Select **Start** in the Data link cart to run the data link job.

6. View the created tables from Database Actions.
   - Click on Database Actions, in the breadcrumb, to go back to the Database Actions launchpad.
   - Click on the SQL tile.
   - Select the external table, drag and drop it into the worksheet.
     The SQL Select statement for the table appears. This SQL statement can be run to consume the shared data.



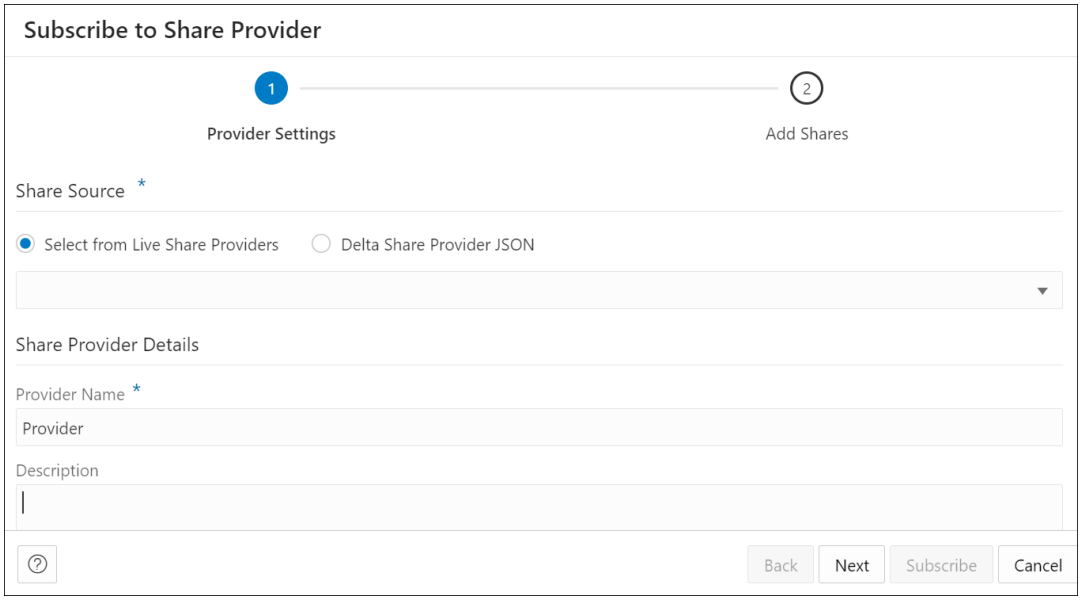
Consume Live Share

Live data shared with you through data sharing is not automatically available for consumption.

To consume live data shares, you need to subscribe to them and create views of tables included in the live share. The views can be queried using SQL scripts.

1. Open the Consume Share page and click **+ Subscribe to Share Provider** to select **Subscribe to Live Share Provider** from the drop-down. This opens the Subscribe to Share Provider dialog box.

2. On the Provider Settings pane of the Subscribe to Share Provider dialog box, specify the following details:



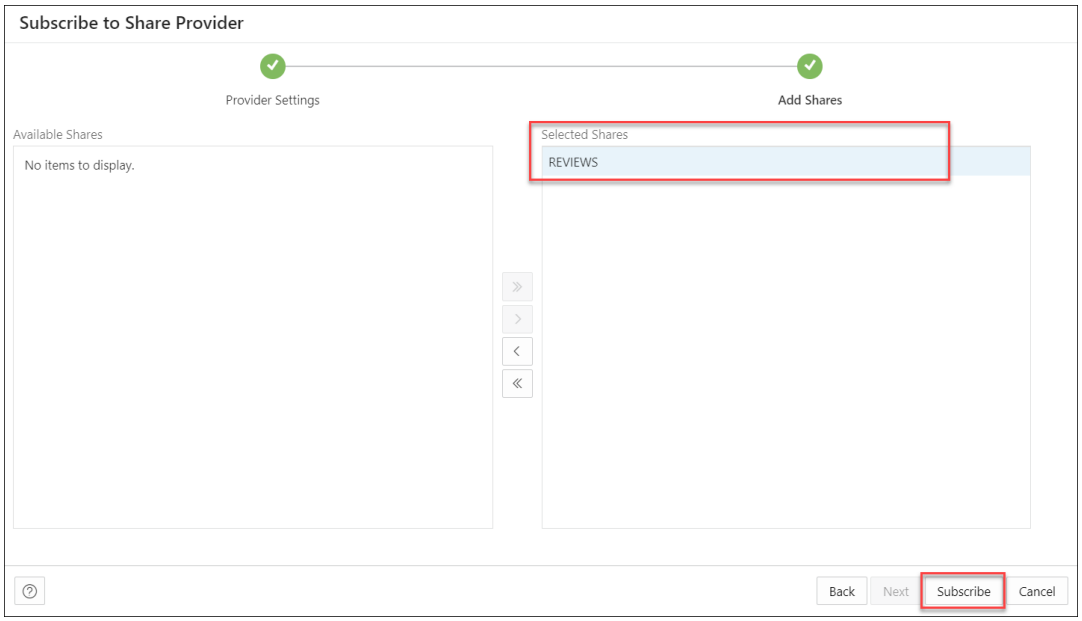
Under Share Source section, choose **Select from Live Share Providers** and select the provider from the drop-down.

Under Share Provider Details field, enter the following:

- **Provider Name**: Specify the name of the provider.
- **Description**: Enter a description of the Provider.

Click **Next** to progress to the Add Shares tab.

3. On the Add Shares tab of the dialog, you will view the list of available shares. Click the share you wish to consume from the Available Shares and select any of the available options:

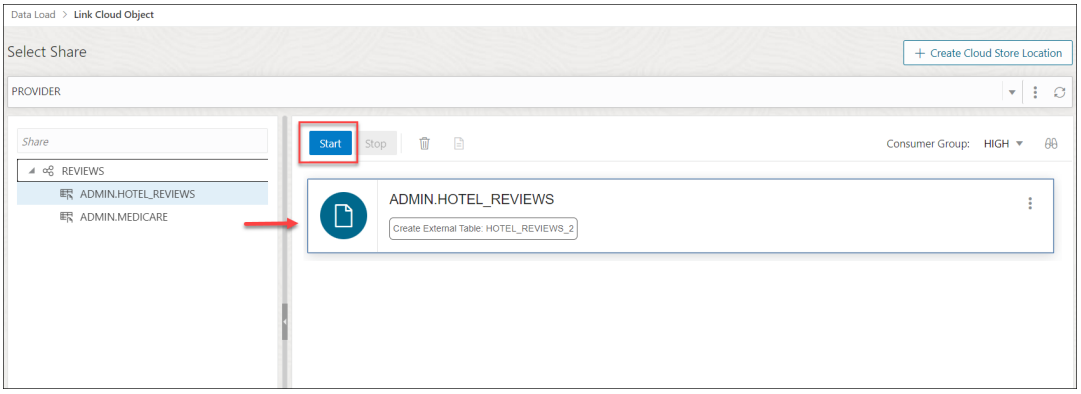


- **>**: This option enables you to move the Available Share to Selected Shares.
- **<**: Select this option to remove the selected share from Selected Shares.
- **>>**: This option allows you to move all the shares to the Selected Shares screen.

- **<<**: Select this option to remove all the selected shares from Selected Shares.

Click **Subscribe** to add the share. A confirmation prompt appears when the provider is created successfully. After successful creation of the provider, you will now view the Link Cloud Object screen of the Data Load page.

4. You can view the name of the Share provider in the cloud storage location field. The share appears in the source file location with the files you add to the share.



Expand the Share folder cart, drag and drop the file you share from the source to the Data Link cart.

Select **Start** in the Data link cart to run the data link job.

View Share Provider Entity Details

To view details about the Share Provider entity, click the **Actions** icon at the right of the Share Provider entity entry, then click **View Details**.

For all entities, the details include Lineage and Impact sections.

For a specific Share Provider entity, you can perform the following actions using the **Actions** context menu.

- **View Details:** See View Share Provider Entity Details.
- **Manage Shares:** Opens a Manage Shares for Share provider dialog box. This lists the shares you select to share with the recipient. You can edit the list of shares you wish to share with the recipient. Click **OK** to save any changes or select **Cancel** to discard the process of editing.
- **Rename:** Renames the Provider Name. Select **Yes** to make changes else click **No**.
- **Delete:** Removes the Share Provider Entity.
- **Load Tables:** You are directed to the Load Data page with the Share tab selected.
- **Link Tables:** You are directed to the Link Data page with the Share tab selected to view and run the related link object. Drag and drop the shared data to add it to the data link job.

Data Share Limitations on Autonomous Database

This section summarizes the limitations of Data Share (both Versioned and Live Share) included in the Oracle Autonomous Database.

Oracle Database Release Restriction

Only **Oracle Database 19c** supports both Versioned Share and Live Share. You cannot use Versioned Share and Live Share with any other database version.

Supported Column Types

Only the following list of column types are supported:

- BINARY_DOUBLE
- BINARY_FLOAT
- CHAR
- CLOB
- DATE
- FLOAT
- INTERVAL DAY TO SECOND
- INTERVAL YEAR TO MONTH
- NCHAR
- NCLOB
- NUMBER
- NVARCHAR2
- TIMESTAMP
- VARCHAR2

Live Share Data Provider

The following are limitations for using the Data Share tool as a Data Share provider:

1. A live share data provider can create shares with a maximum of four objects.
2. If the live share data provider is sharing multiple tables which would normally be joined together, it is recommended that the producer create a view that performs the joins and then only shares the view.

FAQs on the Data Share Tool

The following are frequently asked questions on the Data Share tool.

1. When do recipients receive the Data Definition Language (DDL) updates to the source tables?
   The recipients need to link objects again from the cloud storage location.
   - Navigate to the link cloud Object page from the Data Studio menu.
   - Select the table in the Data Link cart.
   - Click on the **Actions** icon on the table and select Settings.
     This opens the Link Data from Cloud Store Location dialog.
   - Click the **Settings** pane and select "Drop Table and Create New External Table" option from the drop-down.

   You will receive the updated source table.
2. When do recipients receive the Data Manipulation Language (DML) updates to the source tables?
   The provider selects the option to publish versioned data, changes will be reflected if the provider publishes a new version.

Develop

Describes developer tasks and procedures on Autonomous Database.

- Create Applications with Oracle APEX in Autonomous Database
  You can create applications with Oracle APEX on Autonomous Database.
- Using JSON Documents with Autonomous Database
  Autonomous Database has full support for data represented as JSON documents. In Autonomous Databases, JSON documents can coexist with relational data.
- Developing RESTful Services in Autonomous Database
  You can develop and deploy RESTful Services with native Oracle REST Data Services (ORDS) support on Autonomous Databases. Simple Oracle Document Access (SODA) for REST lets you use a database as a simple JSON document store.
- Using Oracle Database API for MongoDB
  Oracle Database API for MongoDB makes it possible to connect to Oracle Autonomous Database using MongoDB language drivers and tools.
- Send Email and Notifications on Autonomous Database
  There are a number of options for sending email on Autonomous Database. You can also send notifications to a Slack or MSTeams channel.
- User Defined Notification Handler for Scheduler Jobs
- Oracle Extensions for IDEs
  Oracle extensions let developers connect to, browse, and manage Autonomous Databases directly from common IDEs.
- Using Oracle Java on Autonomous Database
- Test Workloads with Oracle Real Application Testing
  Oracle Real Application Testing is an extremely cost-effective and easy-to-use proactive performance management solution that enables businesses to fully assess the outcome of a system change in test or production.
- Manage and Store Files in a Cloud Code Repository with Autonomous Database
  Autonomous Database provides routines to manage and store files in Cloud Code (Git) Repositories. The supported Cloud Code Repositories are: GitHub, AWS CodeCommit, and Azure Repos.
- Invoke User Defined Functions
  You can invoke external functions such as OCI, AWS Lambda and Azure remote functions in your Autonomous Database as SQL functions.
- Use Cloud Tables to Store Logging and Diagnostic Information
  You can create Cloud Tables where table data resides on Oracle managed Cloud Storage and the table data does not consume database storage.

Create Applications with Oracle APEX in Autonomous Database

You can create applications with Oracle APEX on Autonomous Database.

- About Oracle APEX
  Oracle APEX is a low-code development platform that enables you to build scalable, secure enterprise applications with world-class features that can be deployed anywhere.
- Access Oracle APEX Administration Services
  Each Autonomous Database instance includes a dedicated instance of Oracle APEX; you can use this instance to create multiple workspaces. A workspace is a shared work area

where you can build applications. You create workspaces in Oracle APEX Administration Services.

- Create Oracle APEX Workspaces in Autonomous Database
  An Autonomous Database instance does not have precreated Oracle APEX workspaces. Create a workspace if you have not already done so or use these instructions to create additional workspaces.
- Access Oracle APEX App Builder
  Use App Builder to create and manage Oracle APEX applications and application pages. The App Builder home page displays all installed applications in the current Oracle APEX workspace.
- Create Oracle APEX Developer Accounts
- Load Data from the Cloud into Oracle APEX
  Using Oracle APEX SQL Workshop, you can declaratively load data from Object Store into the database schema associated with your Oracle APEX workspace.
- Use JSON Data with Oracle APEX
  You can use Oracle APEX to create applications with JSON data. You must first create a view to extract the required attributes from the JSON data and maps them into columns of a relational view.
- Use Web Services with Oracle APEX
- Send Email from Oracle APEX
  You can use the `APEX_MAIL` package to send emails from Oracle APEX applications deployed in Autonomous Database.
- Control Oracle APEX Upgrades
  By default, Autonomous Database applies Oracle APEX upgrades as soon as they are released in your region. You can set an option to defer upgrades for major Oracle APEX releases, such as an upgrade from 22.1 to 23.1, for up to 90 days.
- Access Oracle APEX, Oracle REST Data Services, and Developer Tools Using a Vanity URL
  By default you access Oracle APEX apps, REST endpoints, and developer tools on Autonomous Database using the `oraclecloudapps.com` domain name. You can optionally configure a vanity URL or custom domain name that is easy to remember to help promote your brand identity.
- Oracle APEX Limitations on Autonomous Database
  This section summarizes restrictions and limitations of Oracle APEX included in Oracle Autonomous Database, including for the APEX Service.

About Oracle APEX

Oracle APEX is a low-code development platform that enables you to build scalable, secure enterprise applications with world-class features that can be deployed anywhere.

Oracle APEX provides you with an easy-to-use browser-based environment to load data, manage database objects, develop REST interfaces, and build applications which look and run great on both desktop and mobile devices. You can use Oracle APEX to develop a wide variety of solutions: import spreadsheets and develop a single source of truth in minutes, create compelling data visualizations against your existing data, deploy productivity applications to elegantly solve a business need, or build your next mission-critical data management application.

Oracle APEX embraces SQL. Anything you can express with SQL can be easily employed in an Oracle APEX application. Oracle APEX also embodies low code with powerful data

management and data visualization components, as well as responsive development out of the box. Instead of writing code by hand, you are able to use intelligent wizards to guide you through the rapid creation of applications and components.

Oracle APEX on Autonomous Database provides a preconfigured, fully managed and secured environment to both build and deploy world-class data-centric applications. There are no limits on the number of developers or end users for your Oracle APEX applications; Autonomous Database can instantly scale compute and storage online as needed, based upon your workload. Additionally, Oracle APEX applications developed on-premise can be easily deployed to Oracle APEX on Autonomous Database, or vice-versa.

Configuration, patching, monitoring, and upgrading of all Oracle APEX components is fully managed by Oracle, leaving you free to focus on developing your solutions and solving your business problems. With Oracle APEX and low code, your organization can be more agile and develop solutions faster, for less cost, and with greater consistency. You can adapt to changing requirements with ease. And you can empower professional developers and everyone else in your organization to be a part of the solution.

This chapter covers information on Oracle APEX specific to working on Autonomous Database.

For more information on APEX, see the following:

- Oracle APEX Release 23.2
- apex.oracle.com

Access Oracle APEX Administration Services

Each Autonomous Database instance includes a dedicated instance of Oracle APEX; you can use this instance to create multiple workspaces. A workspace is a shared work area where you can build applications. You create workspaces in Oracle APEX Administration Services.

> **Note:**
>
> If your Autonomous Database is configured to use a Private Endpoint, then you can only access Oracle APEX from clients in the same Virtual Cloud Network (VCN). See Configure Network Access with Private Endpoints for more information.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To access Oracle APEX Administration Services you can use the Oracle Cloud Infrastructure Console or Database Actions.

To access Administration Services from Database Actions:

1. On the Autonomous Database details page select **Database Actions** and in the list click **View all database actions**.

2. From the Database Actions Launchpad, under **Development**, click **APEX**.

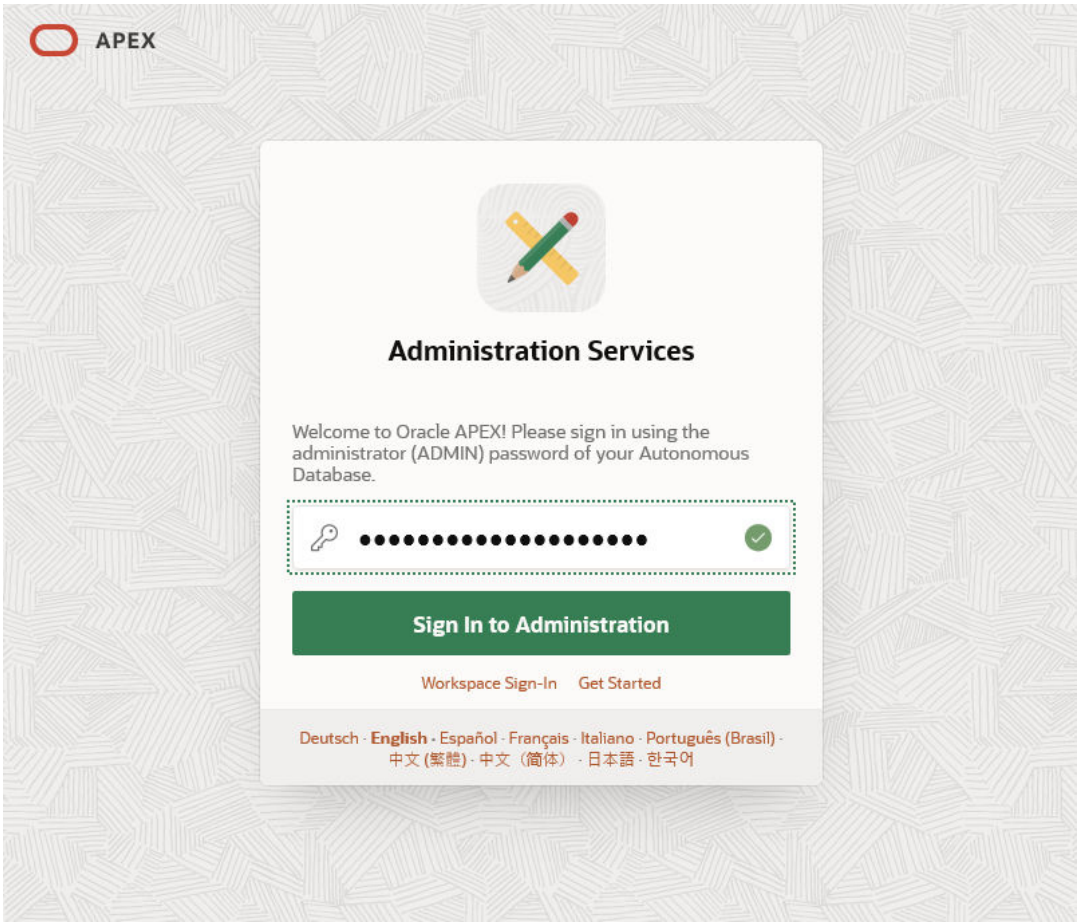
   The Oracle APEX Administration Services sign-in page appears.

   If you already created a workspace, the Application Express workspace sign-in page appears instead. In this case, to open Administration Services, click Administration Services link.

3. In the **Password** field, enter the password for the ADMIN user.

> **Note:**
>
> By default, Administration Services and the Oracle APEX development environment on Autonomous Database use Database Accounts authentication. This authentication method uses the database account user name and password to authenticate users. You can switch to a different authentication method on the Administration Services, Manage Instance, Security page.

See Set the ADMIN Password in Autonomous Database to change the password.



4. Click **Sign In to Administration**.

   When you sign in for the first time, follow the prompts to create an Application Express workspace. See Create Oracle APEX Workspaces in Autonomous Database for more information.

To access Administration Services from Oracle Cloud Infrastructure Console:

1. On the Autonomous Database details page click the **Tools** tab.
2. In the **Tools** column, select the Oracle APEX link.

You can also use Administration Services to manage your APEX instance. See Oracle APEX Administration Services in *Oracle APEX Administration Guide* for more information.

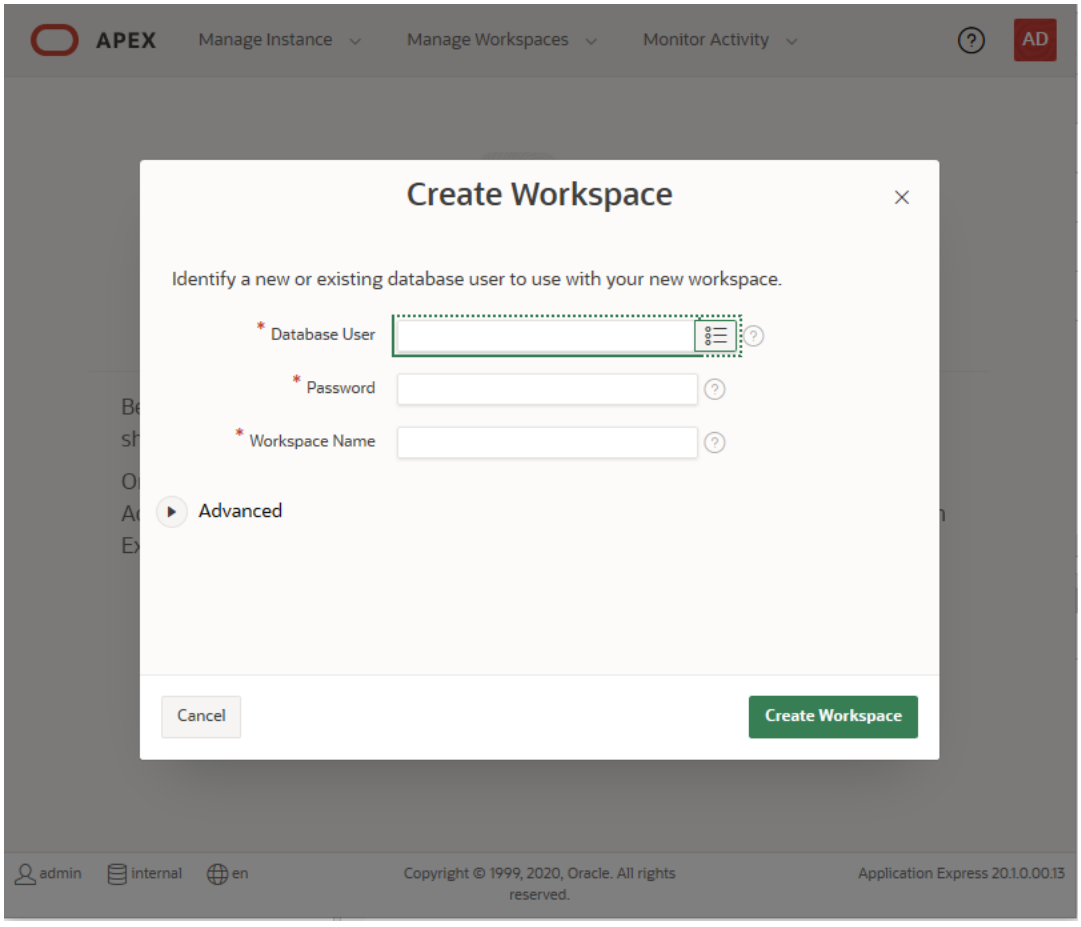
Create Oracle APEX Workspaces in Autonomous Database

An Autonomous Database instance does not have precreated Oracle APEX workspaces. Create a workspace if you have not already done so or use these instructions to create additional workspaces.

To create an Oracle APEX workspace:

1. Sign in to Oracle APEX Administration Services.

   See Access Oracle APEX Administration Services for more information.
2. Click **Create Workspace**.



3. On the Create Workspace page, in the **Database User** field, enter a new database username or choose an existing user from the list.

   The ADMIN database user cannot be associated with a workspace.

4. In the Password field, provide a strong password if the database user is a new user. If the user is an existing database user you do not enter a password.

   See About User Passwords on Autonomous Database to learn more about the default password complexity rules.

5. (optional) In the **Workspace Name** field, change the name of the workspace that was automatically populated.

6. Click **Create Workspace**.

See Access Oracle APEX App Builder and Create Oracle APEX Developer Accounts to create additional developer accounts.

Access Oracle APEX App Builder

Use App Builder to create and manage Oracle APEX applications and application pages. The App Builder home page displays all installed applications in the current Oracle APEX workspace.

If your Autonomous Database is configured to use a Private Endpoint, you can only access Oracle APEX App Builder from clients in the same Virtual Cloud Network (VCN).

See Configure Network Access with Private Endpoints for more information.

To access Oracle APEX App Builder:

1. Sign in to Oracle APEX using the workspace name, username, and password you specify when you create the workspace.

   > **Note:**
   >
   > Oracle APEX Administration Services and the Oracle APEX development environment on Autonomous Database use Database Accounts authentication. This authentication method uses the database account user name and password to authenticate users.

2. On the Workspace home page, click the App Builder icon.

See Create Oracle APEX Developer Accounts to create developer accounts.

Create Oracle APEX Developer Accounts

Oracle APEX developers need a developer account in each workspace where they wish to build applications.

The initial developer account is created when you create a workspace (this account also has Workspace Administrator privilege). These steps show you how to create additional developer accounts for members of your team or reset their passwords. When you create a developer account, a corresponding database user is automatically created.

To create developer accounts and provide direct access to Oracle APEX:

1. Sign in to Oracle APEX Administration Services.

   See Access Oracle APEX Administration Services for more information.

2. Click **Manage Workspaces**.

3. Under Workspace Actions, click **Manage Developers and Users**.

4. On the Manage Application Developers and Users page, click **Create User**.
5. On the Create/Edit User page, in the **Username** field, enter a username.
6. In the **Email Address** field, enter an email address.
7. (Optional) Use the on-screen and in-line help to fill in additional fields.
8. In the **User is an administrator** field, select **No**.
9. In the **User is a developer** field, select **Yes**.
10. In the **Password** field, enter a strong password.

    See Create Users on Autonomous Database - Connecting with a Client Tool to learn more about the default password complexity rules.
11. In the **Confirm Password** field, confirm the password.
12. At the top of the page, click **Create User**.

    Alternatively, click **Create and Create Another** if you want to create the user and create another user.

To share sign-in details with developers:

1. On the Autonomous Databases page, under the **Display Name** column, select the Autonomous Database instance that contains the user and the user's workspace.
2. On the Autonomous Database Details page select **Database Actions** and click **View all database actions**.
3. In Database Actions Launchpad, under **Development**, right-click **APEX** and choose **Copy URL**.
4. Provide the copied URL, along with the Workspace Name, the Username, and the Password for the developer account you created.

Using this URL developers can access the Oracle APEX environment without having to navigate to the Autonomous Database Oracle Cloud Infrastructure Console.

> **Note:**
>
> Changing the password of Workspace Administrators and Developers through **Manage Developers and Users** page or **Edit Profile** page only affects applications configured with "Application Express Accounts" authentication scheme. To change the password used to access App Builder, use Database Actions or another client to change the password of the corresponding database user.

See Creating User Accounts in *Oracle APEX Administration Guide* for more information.

Load Data from the Cloud into Oracle APEX

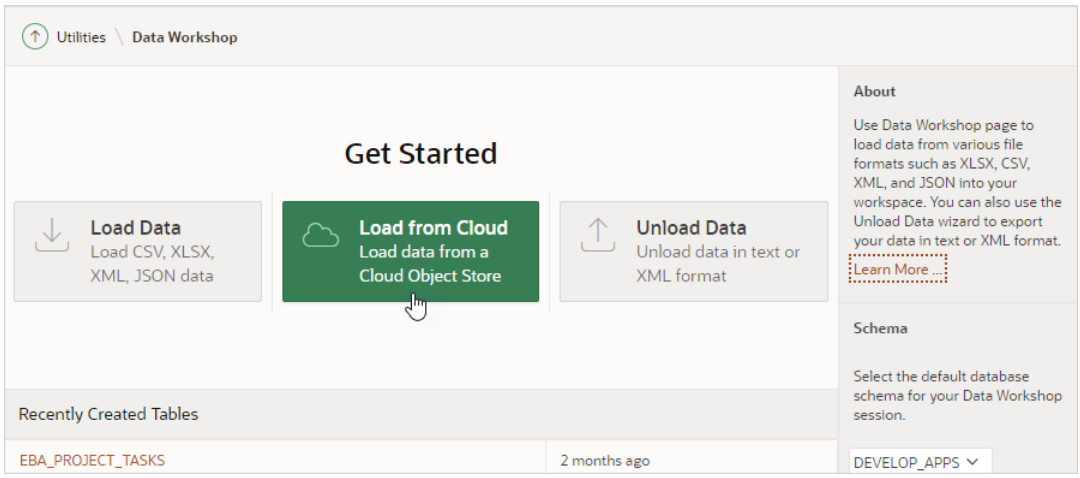
Using Oracle APEX SQL Workshop, you can declaratively load data from Object Store into the database schema associated with your Oracle APEX workspace.

To load data from the cloud, do the following:

1. Open your Oracle APEX workspace.
2. Select **SQL Workshop**.
3. Select **Utilities**.

4. Select **Data Workshop**.

   This shows the Load from Cloud button.



5. Under Get Started, select **Load from Cloud**.

See Loading Data From the Cloud for more information.

Use JSON Data with Oracle APEX

You can use Oracle APEX to create applications with JSON data. You must first create a view to extract the required attributes from the JSON data and maps them into columns of a relational view.

- Create a View from JSON Data Guide
  Oracle APEX interprets data in relational format. Creating a view extracts required attributes from the JSON data and maps them into columns of a relational view.
- Create a View with JSON_TABLE Function
  You can create views of JSON data using the `json_table` SQL/JSON function.

Create a View from JSON Data Guide

Oracle APEX interprets data in relational format. Creating a view extracts required attributes from the JSON data and maps them into columns of a relational view.

For creating a view of JSON data that is stored in SODA collections, you can use SODA APIs and JSON Data Guide. The following PL/SQL code uses SODA APIs to create a Data Guide view on JSON Data stored in SODA Collections.

Run the following code in Oracle APEX SQL Workshop to create a view named `myview`:

```
-- Fetch the data guide and create a view
DECLARE
    coll  SODA_Collection_T;
    dg    CLOB;
    n     NUMBER;
BEGIN    -- Fetch the data guide from the collection or create one with
hierarchical format
    coll := dbms_soda.open_Collection('mycollection');
    dg := coll.get_Data_Guide;
    dbms_output.put_line(JSON_QUERY(dg, '$' pretty));
```

```
    -- User can modify the data guide as needed
    n := coll.create_View_From_DG('myview', dg);
    dbms_output.put_line('Status: ' || n);
    dbms_lob.freeTemporary(dg);
END;
/
```

Use the following command to check if the view has been created:

```
select count(1) from user_views where view_name = 'myview';
```

Use the following command to see the structure of the view:

```
describe myview;
```

See Create View using JSON Data Guide for more information on creating a view using JSON Data Guide.

Create a View with JSON_TABLE Function

You can create views of JSON data using the `json_table` SQL/JSON function.

The `json_table` SQL/JSON function projects specific JSON data to columns of various SQL data types. You can use the `json_table` function to map parts of a JSON document into the rows and columns of a new, virtual table, which you can also think of as an inline view.

See Create View on JSON Data for more information on creating views over JSON Data.

Use Web Services with Oracle APEX

You can interact with both SOAP and RESTful style web services from Oracle APEX in your Autonomous Database instance.
Web services enable applications to interact with one another over the web in a platform-neutral, language independent environment. In a typical web services scenario, a business application sends a request to a service at a given URL by using the HTTP protocol. The service receives the request, processes it, and returns a response. Web services are typically based on Simple Object Access Protocol (SOAP) or Representational State Transfer (REST) architectures.

Using REST Data Sources (formerly called Web Source Modules), APEX developers can declaratively access data services from a variety of REST endpoints, allowing both read and write operations. In addition to supporting smart caching rules for remote REST data, Oracle APEX also offers the unique ability to directly manipulate the results of REST data sources using industry standard SQL.

The `APEX_WEB_SERVICE` package enables you to integrate other systems with APEX by allowing you to interact with web services anywhere you can use PL/SQL in your application. The package contains procedures and functions to call both SOAP and RESTful style web services, and to simplify implementation of OAuth 2.0 flows.

Note the following when working with web services in APEX with Autonomous Database:

- All web services must be secured. Only HTTPS services are supported on the default port (443). Connections through IP addresses are not allowed. Your APEX instance is preconfigured with an Oracle Wallet that contains more than 90 of the most common trusted root and intermediate SSL certificates. The `APEX_WEB_SERVICE` package

automatically takes advantage of this Oracle Wallet without additional configuration from application developers.

- Each Autonomous Database instance is preconfigured with a network access control list (ACL) to permit outbound web service calls from Oracle APEX to public endpoints.
- In order to reach endpoints in private subnets or behind on-premises firewalls, ensure they meet the prerequisites from this section, Submit an HTTP Request to a Private Host with UTL_HTTP, and add the following access control list for the desired host:

```
BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
       host => 'www.example.com',
       ace => XS$ACE_TYPE(
           privilege_list => XS$NAME_LIST('http'),
           principal_name => APEX_APPLICATION.g_flow_schema_owner,
           principal_type => XS_ACL.ptype_db),
       private_target => true);
END;
/
```

> **Note:**
>
> If you set `ROUTE_OUTBOUND_CONNECTIONS` database property to `PRIVATE_ENDPOINT`, you do not need to define access control lists for individual hosts in order to access them from APEX. Enhanced Security for Outbound Connections with Private Endpoints

- Your APEX instance does not require an outbound web proxy. Attempting to use a web proxy at the API call level or application level shows the error: `ORA-01031: insufficient privileges`.
- There is a default limit of 50,000 outbound web service requests per APEX workspace in a rolling 24-hour period. If the limit of outbound web service calls is reached, the following SQL exception is raised on the subsequent request and the request is blocked:

  ```
  ORA-20001: You have exceeded the maximum number of web service requests per
  workspace. Please contact your administrator.
  ```

  You can raise or remove the default limit of outbound web service requests by setting a value for the `MAX_WEBSERVICE_REQUESTS` instance parameter or by updating the Maximum Web Service Requests attribute in APEX Administration Services. For example, to change the limit to 250,000, connect to your database as ADMIN using a SQL client and execute the following:

  ```
  BEGIN
      APEX_INSTANCE_ADMIN.SET_PARAMETER('MAX_WEBSERVICE_REQUESTS', '250000');
      COMMIT;
  END;
  /
  ```

To learn more, see:

- APEX_WEB_SERVICE in *Oracle APEX API Reference*
- Managing REST Data Sources in *Oracle APEX App Builder User's Guide*

Send Email from Oracle APEX

You can use the `APEX_MAIL` package to send emails from Oracle APEX applications deployed in Autonomous Database.

Before you use `APEX_MAIL` you must configure an email provider in your Oracle APEX instance. The only supported email provider is *Oracle Cloud Infrastructure Email Delivery* service.

> **Note:**
>
> Third-party email providers are not supported.

To enable `APEX_MAIL` functionality in your APEX instance in Autonomous Database:

1. Identify the SMTP connection endpoint for Email Delivery. You configure the endpoint as the SMTP Host in your APEX instance in Step 4. You may need to subscribe to additional Oracle Cloud Infrastructure regions if Email Delivery is not available in your current region. See Configure SMTP Connection for more information.
2. Generate SMTP credentials for Email Delivery. Your APEX instance uses credentials to authenticate with Email Delivery servers when you send email. See Generate SMTP Credentials for a User for more information.
3. Create an approved sender for Email Delivery. You need to complete this step for all email addresses you use as the "From" with `APEX_MAIL.SEND` calls, as the Application Email From Address in your apps, or in the `SMTP_FROM` instance parameter. See Managing Approved Senders for more information.
4. Connect to your Autonomous Database as ADMIN user using a SQL client and configure the following SMTP parameters using `APEX_INSTANCE_ADMIN.SET_PARAMETER`:
   - `SMTP_HOST_ADDRESS`: Specifies the SMTP connection endpoint from Step 1.
   - `SMTP_USERNAME`: Specifies the SMTP credential user name from Step 2.
   - `SMTP_PASSWORD`: Specifies the SMTP credential password from Step 2.
   - Keep default values for `SMTP_HOST_PORT` parameter (587) and `SMTP_TLS_MODE` parameter (`STARTTLS`).

   For example:

   ```
   BEGIN
       APEX_INSTANCE_ADMIN.SET_PARAMETER('SMTP_HOST_ADDRESS', 'smtp.us-
   phoenix-1.oraclecloud.com');
       APEX_INSTANCE_ADMIN.SET_PARAMETER('SMTP_USERNAME',
   'ocid1.user.oc1.username');
       APEX_INSTANCE_ADMIN.SET_PARAMETER('SMTP_PASSWORD', 'password');
       COMMIT;
   END;
   /
   ```

5. Validate the email configuration settings using a SQL client.

   ```
   BEGIN
       APEX_INSTANCE_ADMIN.VALIDATE_EMAIL_CONFIG;
   ```

```
END;
/
```

If any errors are reported (for example, "`ORA-29279: SMTP permanent error: 535 Authentication credentials invalid`"), adjust the SMTP parameters and repeat the validation step.

6. Send a test email using APEX SQL Workshop, SQL Commands specifying one of the approved senders from Step 3 as "From". For example:

```
BEGIN
    APEX_MAIL.SEND(p_from => 'alice@example.com',
                   p_to   => 'bob@example.com',
                   p_subj => 'Email from Oracle Autonomous Database',
                   p_body => 'Sent using APEX_MAIL');
END;
/
```

7. To monitor email delivery in your APEX instance:
   a. Sign in to APEX Administration Services.
   b. Open the Manage Instance page.
   c. Click the Mail Queue link in the Manage Meta Data section.

   Alternatively, query `APEX_MAIL_QUEUE` and `APEX_MAIL_LOG` views using a SQL client.

> **Note:**
>
> There is a default limit of 5,000 emails per workspace in a 24-hour period. You can update or remove this limit in Oracle APEX Administration Services or by setting the `WORKSPACE_EMAIL_MAXIMUM` instance parameter. Oracle Cloud Infrastructure Email Delivery may impose additional limitations.

An approved sender must be set up for all “From:” addresses sending mail though Oracle Cloud Infrastructure, or mail will be rejected. There are limitations on approved senders with Oracle Cloud Infrastructure Email Delivery. See Managing Approved Senders for more information.

For more information, see:

- Overview of the Email Delivery Service
- APEX_MAIL in *Oracle APEX API Reference*
- APEX_INSTANCE_ADMIN in *Oracle APEX API Reference*

Control Oracle APEX Upgrades

By default, Autonomous Database applies Oracle APEX upgrades as soon as they are released in your region. You can set an option to defer upgrades for major Oracle APEX releases, such as an upgrade from 22.1 to 23.1, for up to 90 days.

> **Note:**
>
> The latest Oracle APEX Patch Set Bundles are automatically applied on Autonomous Database and cannot be deferred.

- Defer Oracle APEX Upgrades
  Describes the steps to defer Oracle APEX upgrades.
- Apply Oracle APEX Upgrades
  When you have deferred upgrades and an upgrade is available, you have the option to apply the upgrade at any time before the specified upgrade date.

Defer Oracle APEX Upgrades

Describes the steps to defer Oracle APEX upgrades.

Deferring Oracle APEX upgrades allows you to do the following:

- Control when Oracle APEX is upgraded on your Autonomous Database. While there should be minimal disruption on a live system, you can still choose to upgrade at a time or on a day with less activity, for example on a weekend day.
- Validate your applications against the new version of Oracle APEX before you upgrade your production system.

  To validate and test before you upgrade your production system, do the following:

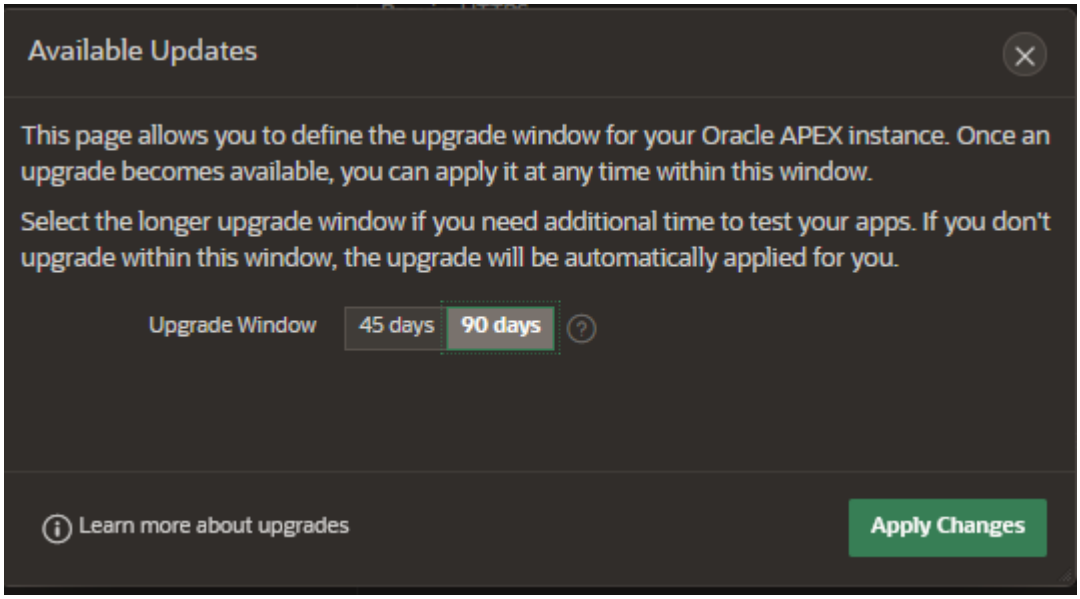
  – Clone your production or test environment. See Clone an Autonomous Database Instance for more information.
  – Upgrade Oracle APEX on the clone as specified in Apply Oracle APEX Upgrades.
  – Perform your testing and validation on the clone in a non-production environment.

To defer Oracle APEX upgrades:

1. Open Oracle APEX Administration Services.

   See Access Oracle APEX Administration Services for more information.
2. Click **APEX** to open the Administration Services page.
3. Open the Available Updates dialog by clicking the  next to **Available Updates**.

   This shows the Available Updates dialog.
4. In the Available Updates dialog select an **Upgrade Window**.



5. Click **Apply Changes**.

Notes for deferred Oracle APEX upgrades:

- If an Upgrade Window is selected and updates are available, then on the date indicated in the Available Updates dialog, Oracle APEX updates are automatically applied to your Autonomous Database instance. You cannot postpone the updates beyond the listed date.
- If an upgrade window is set and updates are available, you cannot schedule the Oracle APEX upgrades for a specific date and time. When you are ready to upgrade, access Oracle APEX Administration Services and click **Upgrade Now**. See Apply Oracle APEX Upgrades for details.
- If your Autonomous Database is stopped at the time when a new Oracle APEX release becomes available in your region, the update is automatically applied the next time you start your database, regardless of the Upgrade Window setting. For example, if the Autonomous Database is stopped when a new Oracle APEX release becomes available, based on the Auto Start/Stop Schedule, the update is automatically applied the next time the database starts.
- When you change the Upgrade Window setting, the change applies to future upgrades and does not impact an Oracle APEX upgrade that is already available to your Autonomous Database.
- If you configure Customer Managed Oracle REST Data Services (ORDS) in your Autonomous Database instance, APEX upgrades are always deferred, regardless of the **Upgrade Window** setting. See Apply Oracle APEX Upgrades to apply an available APEX upgrade with Customer Managed Oracle REST Data Services (ORDS).

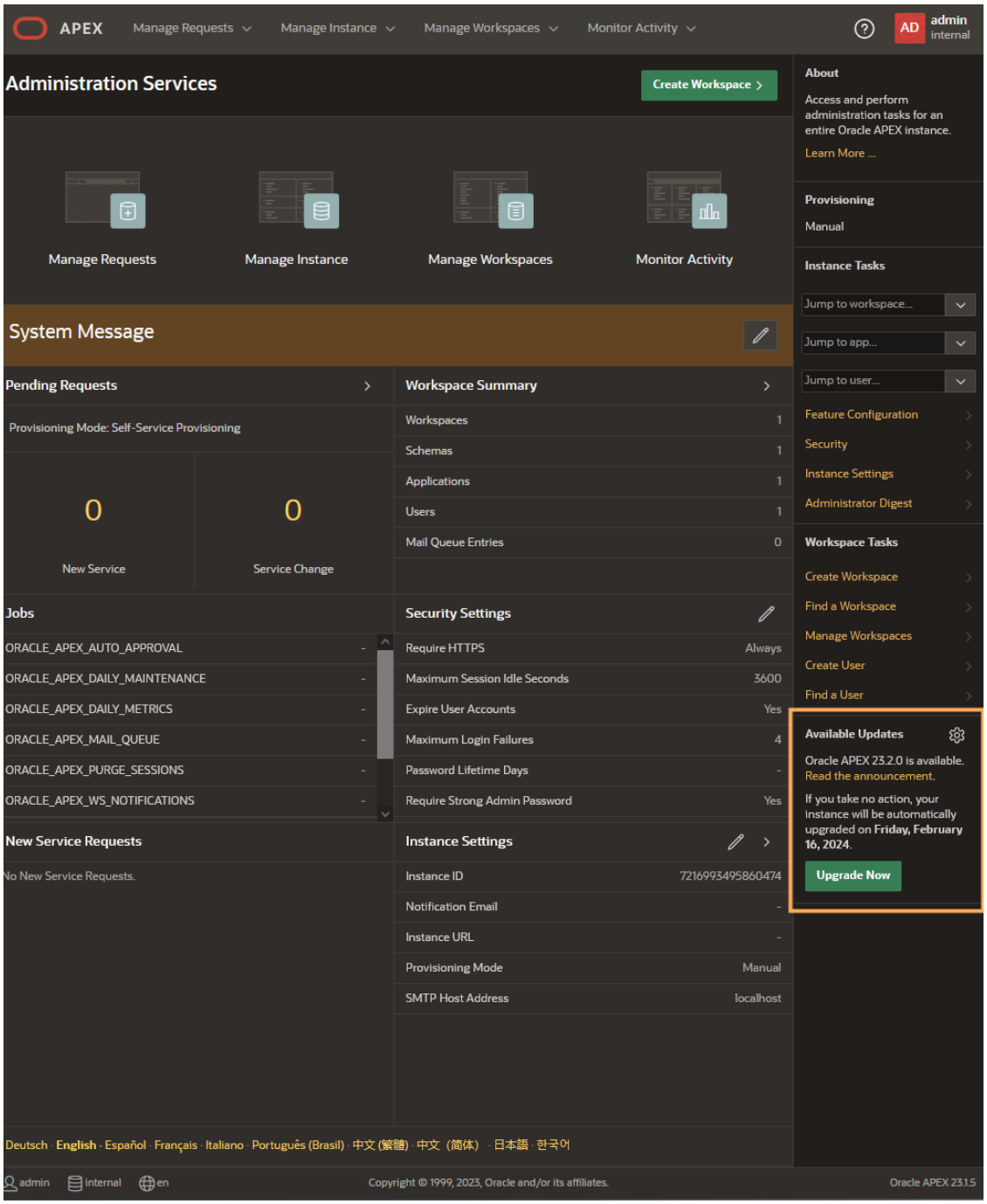
  See About Customer Managed Oracle REST Data Services on Autonomous Database for more information.

Apply Oracle APEX Upgrades

When you have deferred upgrades and an upgrade is available, you have the option to apply the upgrade at any time before the specified upgrade date.

When Oracle APEX upgrades are not available the Available Updates area shows **System is up-to-date**.

When the Defer Upgrade setting specifies an Upgrade Window, if an Oracle APEX upgrade is available and you take no action, the upgrade is applied automatically on the date specified in the Available Updates area.



If you previously deferred upgrades by specifying an upgrade window, as specified in Defer Oracle APEX Upgrades, then you can upgrade at any time, as follows:

1. Open Oracle APEX Administration Services.
2. Click **APEX** to open the Administration Services page.
3. Open the Available Updates dialog by clicking the  next to **Available Updates**.
4. On the Available Updates dialog click **Upgrade Now**.

This starts the Oracle APEX upgrade process in the background. You can continue using Administration Services and App Builder as well as running apps during most of the upgrade process.

Access Oracle APEX, Oracle REST Data Services, and Developer Tools Using a Vanity URL

By default you access Oracle APEX apps, REST endpoints, and developer tools on Autonomous Database using the `oraclecloudapps.com` domain name. You can optionally configure a vanity URL or custom domain name that is easy to remember to help promote your brand identity.

After you acquire a desired domain name and matching SSL certificate from a vendor of your choice, deploy an Oracle Cloud Infrastructure Load Balancer in your Virtual Cloud Network (VCN) using your Autonomous Database as the backend. Your Autonomous Database instance must be configured with a private endpoint in the same VCN. See Configuring Network Access with Private Endpoints for more information.

To learn more, see the following:

- Introducing Vanity URLs for APEX and ORDS on Oracle Autonomous Database
- Automate Vanity URL Configuration Using Terraform

Oracle APEX Limitations on Autonomous Database

This section summarizes restrictions and limitations of Oracle APEX included in Oracle Autonomous Database, including for the APEX Service.

> **Tip:**
>
> This section does not apply to co-managed databases (for example, OCI Base Database Service) and Oracle Autonomous Database on Dedicated Exadata Infrastructure.

In a fully managed environment (such as Oracle Autonomous Database Serverless), certain limitations are required to protect the security and performance of your Oracle APEX environment.

- Administration Services - The following Oracle APEX Administration Services configuration options are disabled, or have been predefined by Oracle and cannot be altered.
  - Manage Instance, Feature Configuration:
    - Monitoring - Web Service Activity Logging, Enable Application Tracing
  - Manage Instance, Security, Security Settings:
    - Security - Instance Proxy, Instance No Proxy Domains, Unhandled Errors
    - Authentication Control:
      - HTTP Protocol - Require Outbound HTTPS
      - General - Single Sign-On Logout URL
  - Manage Instance, Manage Logs and Files:
    - SQL Workshop Log

        - Page View Activity Log
        - Developer Activity Log
        - External Click Counting Log
        - Login Access Log
        - Web Service Activity Log
        - REST Synchronization Log
        - Automation Log
    - Manage Instance, Instance Settings:
        - Storage - All tablespace settings
        - Wallet
    - Manage Instance, Workspace Purge Settings
    - Manage Workspaces, Existing Workspaces, Edit Workspace Information:
        - Login Control
        - Session Timeout
        - Workspace Isolation
- Only the following `APEX_INSTANCE_ADMIN` procedures and functions are supported:
    - `ADD_AUTHORIZED_URL`
    - `ADD_SCHEMA`
    - `ADD_WORKSPACE`
    - `CREATE_OR_UPDATE_ADMIN_USER`
    - `DISABLE_WORKSPACE`
    - `ENABLE_WORKSPACE`
    - `FREE_WORKSPACE_APP_IDS`
    - `GET_AUTHORIZED_URLS`
    - `GET_PARAMETER`
    - `GET_SCHEMAS`
    - `REMOVE_APPLICATION`
    - `REMOVE_AUTHORIZED_URL`
    - `REMOVE_SAVED_REPORTS`
    - `REMOVE_SAVED_REPORT`
    - `REMOVE_SCHEMA`
    - `REMOVE_SUBSCRIPTION`
    - `REMOVE_WORKSPACE`
    - `RESERVE_WORKSPACE_APP_IDS`
    - `SET_LOG_SWITCH_INTERVAL`
    - `SET_PARAMETER`
    - `UNLOCK_USER`

  - `VALIDATE_EMAIL_CONFIG`

  See APEX_INSTANCE_ADMIN in *Oracle APEX API Reference*.

- The following application authentication schemes are supported with limitations:
  - HTTP Header Variable: Only after configuring a vanity URL.

    See Access Oracle APEX, Oracle REST Data Services, and Developer Tools Using a Vanity URL for more information.
  - LDAP Directory, including `APEX_LDAP` API: With the same restrictions that apply to the `DBMS_LDAP` package.

    See PL/SQL Packages Notes for Autonomous Database for more information.
  - SAML Sign-In: Only when using a customer managed ORDS.

    See About Customer Managed Oracle REST Data Services on Autonomous Database for more information.
- Disabled options in SQL Workshop:

  The ability to create and manage database links in Object Browser is disabled. To create a database link, use the `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` PL/SQL API.

  See Use Database Links with Autonomous Database for more information.
- Runtime environment:

  Oracle APEX is only available as a Full Development environment. Converting into a Runtime environment, which minimizes the installed software footprint and removes UI components such as App Builder and Administration Services, is not supported.

  See About the Differences Between Runtime and Full Development Environments in *Oracle APEX App Builder User's Guide* to learn more.

Using JSON Documents with Autonomous Database

Autonomous Database has full support for data represented as JSON documents. In Autonomous Databases, JSON documents can coexist with relational data.

You can access and manage JSON documents as follows:

- Using Simple Oracle Document Access (SODA) APIs for NoSQL-style access to document collections.
- Using SQL and PL/SQL to access JSON documents in tables.
- Work with Simple Oracle Document Access (SODA) in Autonomous Database
  Simple Oracle Document Access (SODA) is a set of NoSQL-style APIs that let you use collections of JSON documents in Autonomous Database, retrieve them, and query them, without needing to know Structured Query Language (SQL) or how the documents are stored in the database.
- Work with JSON Documents Using SQL and PL/SQL APIs on Autonomous Database
  Autonomous Database supports JavaScript Object Notation (JSON) data natively in the database.
- Load JSON Documents with Autonomous Database
  You can use the PL/SQL procedure `DBMS_CLOUD.COPY_COLLECTION` to load JSON documents into SODA collections or use `DBMS_CLOUD.COPY_DATA` to load JSON documents into an existing table in Autonomous Database.

- Import SODA Collection Data Using Oracle Data Pump Version 19.6 or Later
  Shows the steps to import SODA collections into Autonomous Database with Oracle Data Pump.

Work with Simple Oracle Document Access (SODA) in Autonomous Database

Simple Oracle Document Access (SODA) is a set of NoSQL-style APIs that let you use collections of JSON documents in Autonomous Database, retrieve them, and query them, without needing to know Structured Query Language (SQL) or how the documents are stored in the database.

Autonomous Database supports storing and querying JSON documents natively. SODA document collections are backed by ordinary database tables and views; you can take advantage of database features for use with the content of SODA documents.

SODA drivers are available for several languages and frameworks including: Java, Node.js, Python, C (using Oracle Call Interface), and PL/SQL, and SODA for REST. SODA for REST maps SODA operations to Uniform Resource Locator (URL) patterns, so it can be used with most programming languages.

To get started with SODA, see the following:

- Overview of SODA

Depending on the SODA API you want to work with, see the following:

> **Note:**
>
> If you are using Always Free Autonomous Database with Oracle Database 21c, then to avoid compatibility problems of SODA drivers, Oracle recommends the following:
>
> - Use the driver versions that are needed for working with `JSON` type as specified in SODA Drivers. See SODA Drivers for more information.
> - For projects that were started using a database release prior to Oracle Database 21c, explicitly specify the metadata for the default collection as specified in the example in SODA Drivers. For projects started using release Oracle Database 21c or later, just use the default metadata. See SODA Drivers for more information.

SODA API	Download and Installation	More Information
SODA for Java	Download SODA for Java SODA for Java Prerequisites **Versions:** SODA for Java, using the latest version is recommended. The minimum supported version is: 1.1.4. Use SODA for Java in conjunction with `ojdbc8.jar` for 19.6 (available at Oracle Database 19c (19.6) JDBC Driver & UCP Downloads, or on Maven Central). Autonomous Database does not support Metadata builder. To customize collection metadata pass collection metadata strings directly to the createCollection method. See SODA Collection Metadata on Autonomous Database for more information.	SODA for Java
SODA for REST	Access RESTful Services and SODA for REST	Use SODA for REST with Autonomous Database
SODA for C	Oracle Instant Client Downloads **Versions:** For SODA for C, Oracle Client libraries must be 19.6 and above. You can obtain Oracle Instant Client from Oracle Instant Client Downloads.	SODA for C
SODA for PL/SQL	No need to download. This is included with Autonomous Database.	SODA for PL/SQL
SODA for Node.js	Multiple downloads described in install instructions: Quick Start node-oracledb Installation **Versions:** SODA support was introduced in version 3.0. Using the latest version is recommended, the minimum recommended version is 4.0. Oracle Client libraries must be 19.6 and above. You can obtain Oracle Instant Client from Oracle Instant Client Downloads.	Node-oracledb SODA Requirements
SODA for Python	Multiple downloads described in install instructions: cx_Oracle 7 Installation **Versions:** SODA support was introduced in version 7.0. Using the latest version is recommended, the minimum recommended version is 7.1. Oracle Client libraries must be 19.6 and above. You can obtain Oracle Instant Client from Oracle Instant Client Downloads.	Introduction to cx_Oracle

- SODA Notes
  When you use SODA with Autonomous Database the following restrictions apply:

SODA Notes

When you use SODA with Autonomous Database the following restrictions apply:

- Automatic indexing is not supported for SQL and PL/SQL code that uses the SQL/JSON function `json_exists`. See SQL/JSON Condition JSON_EXISTS for more information.
- Automatic indexing is not supported for SODA query-by-example (QBE).

Work with JSON Documents Using SQL and PL/SQL APIs on Autonomous Database

Autonomous Database supports JavaScript Object Notation (JSON) data natively in the database.

When you use Autonomous Database to store JSON data you can take advantage of all the features available in your database. You can combine your JSON data with non-JSON data; then, using Autonomous Database features such as Oracle Machine Learning Notebooks, you can analyze your data and create reports. You can access JSON data stored in the database the same way you access other database data, including using Oracle Call Interface (OCI), Microsoft .NET Framework, and Java Database Connectivity (JDBC).

See JSON in Oracle Database to get started.

Load JSON Documents with Autonomous Database

You can use the PL/SQL procedure `DBMS_CLOUD.COPY_COLLECTION` to load JSON documents into SODA collections or use `DBMS_CLOUD.COPY_DATA` to load JSON documents into an existing table in Autonomous Database.

Topics

- About Loading JSON Documents
- Load a JSON File of Line-Delimited Documents into a Collection
- Load an Array of JSON Documents into a Collection
- Monitor and Troubleshoot COPY_COLLECTION Loads
- Create Credentials and Copy JSON Data into an Existing Table

Import SODA Collection Data Using Oracle Data Pump Version 19.6 or Later

Shows the steps to import SODA collections into Autonomous Database with Oracle Data Pump.

You can export and import SODA collections using Oracle Data Pump Utilities starting with version 19.6. Oracle recommends using the latest Oracle Data Pump version for importing data from Data Pump files into your database.
Download the latest version of Oracle Instant Client, which includes Oracle Data Pump, for your platform from Oracle Instant Client Downloads. See the installation instructions on the platform install download page for the installation steps required after you download Oracle Instant Client.

In Oracle Data Pump, if your source files reside on Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure native URIs, Swift URIs, or pre-authenticated URIs. See DBMS_CLOUD Package File URI Formats for details on these file URI formats.

If you are using an Oracle Cloud Infrastructure pre-authenticated URI, you still need to supply a `credential` parameter. However, credentials for a pre-authenticated URL are ignored (and the supplied credentials do not need to be valid). See DBMS_CLOUD Package File URI Formats for information on Oracle Cloud Infrastructure pre-authenticated URIs.

This example shows how to create the SODA collection metadata and import a SODA collection with Data Pump.

1. On the source database, export the SODA collection using the Oracle Data Pump `expdp` command.

   See Export Your Existing Oracle Database to Import into Autonomous Database for more information.

2. Upload the dump file set from Step 1 to Cloud Object Storage.

3. Create a SODA collection with the required SODA collection metadata on your Autonomous Database.

   For example, if you export a collection named *MyCollectionName* from the source database with the following metadata:

   - The content column is a `BLOB` type.
   - The version column uses the `SHA256` method.

   Then on the Autonomous Database where you import the collection, create a new collection:

   - By default on Autonomous Database for a new collection the content column is set to `BLOB` with the `jsonFormat` specified as `OSON`.
   - By default on Autonomous Database for a new collection the `versionColumn.method` is set to `UUID`.

   See SODA Default Collection Metadata on Autonomous Database for details.

   For example:

```
DECLARE
   collection_create SODA_COLLECTION_T;
BEGIN
   collection_create := DBMS_SODA.CREATE_COLLECTION('MyCollectionName');
END;
/
COMMIT;
```

   You can use the PL/SQL function `DBMS_SODA.LIST_COLLECTION_NAMES` to discover existing collections. See LIST_COLLECTION_NAMES Function for more information.

   You can view the metadata for the SODA collections by querying the view `USER_SODA_COLLECTIONS`. See USER_SODA_COLLECTIONS for more information.

4. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example, to create Oracle Cloud Infrastructure Auth Token credentials:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adb_user@example.com',
```

```
    password => 'password'
  );
END;
/
```

For more information on Oracle Cloud Infrastructure Auth Token authentication see CREATE_CREDENTIAL Procedure.

For example, to create Oracle Cloud Infrastructure Signing Key based credentials:

```
BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL (
       credential_name => 'DEF_CRED_NAME',
       user_ocid       =>
'ocid1.user.oc1..aaaaaaaauq54mi7zdyfhw33ozkwuontjceel7fok5nq3bf2vwetkpqsoa'
,
       tenancy_ocid    =>
'ocid1.tenancy.oc1..aabbbbbbaafcue47pqmrf4vigneebgbcmmoy5r7xvoypicjqqge32ew
nrcyx2a',
       private_key     =>
'MIIEogIBAAKCAQEAtUnxbmrekwgVac6FdWeRzoXvIpA9+0r1.....wtnNpESQQQ0QLGPD8NM//
JEBg=',
       fingerprint     =>
'f2:db:f9:18:a4:aa:fc:94:f4:f6:6c:39:96:16:aa:27');
END;
/
```

For more information on Oracle Cloud Infrastructure Signing Key based credentials see CREATE_CREDENTIAL Procedure.

Supported credential types:

- Data Pump Import supports Oracle Cloud Infrastructure Auth Token based credentials and Oracle Cloud Infrastructure Signing Key based credentials.

  For more information on Oracle Cloud Infrastructure Signing Key based credentials see CREATE_CREDENTIAL Procedure.

- Data Pump supports using an Oracle Cloud Infrastructure Object Storage pre-authenticated URL for the `dumpfile` parameter. When you use a pre-authenticated URL, providing the `credential` parameter is required and `impdp` ignores the `credential` parameter. When you use a pre-authenticated URL for the `dumpfile`, you can use a `NULL` value for the `credential` in the next step. See Using Pre-Authenticated Requests for more information.

5. Run Data Pump Import with the *dumpfile* parameter set to the list of file URLs on your Cloud Object Storage and the *credential* parameter set to the name of the credential you created in the previous step.

> **Note:**
>
> Import the collection data using the option `CONTENT=DATA_ONLY`.

Specify the collection you want to import using the `INCLUDE` parameter. This is useful if a data file set contains the entire schema and the SODA collection you need to import is included as part of the dump file set.

Use `REMAP_DATA` to change any of the columns during import. This example shows using `REMAP_DATA` to change the version column method from `SHA256` to `UUID`.

```
impdp admin/password@db2022adb_high \
     directory=data_pump_dir \
     credential=def_cred_name \
     dumpfile= https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/export%u.dmp \

     encryption_pwd_prompt=yes \
     SCHEMA=my_schema \
     INCLUDE=TABLE:\"= \'MyCollectionName\'\"  \
     CONTENT=DATA_ONLY \
     REMAP_DATA=my_schema.'\"MyCollectionName\"'.VERSION:SYS.DBMS_SODA.TO_UUID
```

> **Note:**
>
> If during the export with `expdp` you used the `encryption_pwd_prompt=yes` parameter then use `encryption_pwd_prompt=yes` and input the same password at the `impdp` prompt that you specified during the export.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

In Oracle Data Pump version 19.6 and later, the *credential* argument authenticates Oracle Data Pump to the Cloud Object Storage service you are using for your source files. The `credential` parameter cannot be an Azure service principal, Amazon Resource Name (ARN), or a Google service account. See Accessing Cloud Resources by Configuring Policies and Roles for more information on resource principal based authentication.

The *dumpfile* argument is a comma delimited list of URLs for your Data Pump files.

For the best import performance use the `HIGH` database service for your import connection and set the `parallel` parameter to one quarter the number of ECPUs (.25 x **ECPU count**). If you are using OCPU compute model, set the parallel parameter to the number of OCPUs (1 x **OCPU count**).

For information on which database service name to connect to run Data Pump Import, see Manage Concurrency and Priorities on Autonomous Database.

For the dump file URL format for different Cloud Object Storage services, see DBMS_CLOUD Package File URI Formats.

> **Note:**
>
> To perform a full import or to import objects that are owned by other users, you need the `DATAPUMP_CLOUD_IMP` role.

For information on disallowed objects in Autonomous Database, see SQL Commands.

In this import example, the specification for the `REMAP_DATA` parameter uses the function `DBMS_SODA.TO_UUID` to generate UUID values. By default, for on-premise databases, the version column of a SODA collection is computed using SHA-256 hash of the document's

content. On Autonomous Database the version column uses UUID generated values, which are independent of the document's content.

In this example the `REMAP_DATA` parameter uses the `DBMS_SODA.TO_UUID` function to replace the source collection version type with UUID versioning. If in the export dump file set that you are importing the `versionColumn.method` is already set to UUID, then the `REMAP_DATA` for this field is not required.

For detailed information on Oracle Data Pump Import parameters see *Oracle Database Utilities*.

The log files for Data Pump Import operations are stored in the directory you specify with the Data Pump Import `DIRECTORY` parameter. See Accessing the Log File for Data Pump Import for more information.

Developing RESTful Services in Autonomous Database

You can develop and deploy RESTful Services with native Oracle REST Data Services (ORDS) support on Autonomous Databases. Simple Oracle Document Access (SODA) for REST lets you use a database as a simple JSON document store.

- About Oracle REST Data Services in Autonomous Database
- Access RESTful Services and SODA for REST
  Each Autonomous Database includes Oracle REST Data Services (ORDS) that provides HTTPS interfaces for working with the contents of your Oracle Database in REST enabled schemas.
- Develop with Oracle REST Data Services on Autonomous Database
  Autonomous Database supports Oracle REST Data Services (ORDS).
- Use SODA for REST with Autonomous Database
  Autonomous Database supports Simple Oracle Document Access (SODA) for REST.
- Access Oracle REST Data Services, Oracle APEX, and Developer Tools Using a Vanity URL
- About Customer Managed Oracle REST Data Services on Autonomous Database

About Oracle REST Data Services in Autonomous Database

Oracle REST Data Services (ORDS) makes it easy to develop REST interfaces for relational data in a database.

ORDS is a mid-tier Java application that maps HTTP(S) verbs, such as GET, POST, PUT, DELETE, and so on, to database transactions, and returns any results as JSON data.

> **Note:**
>
> The Oracle REST Data Services (ORDS) application in Autonomous Database is preconfigured and fully managed. ORDS connects to the database using the `low` predefined database service with a fixed maximum number of connections (the number of connections for ORDS does not change based on the number of ECPUs (OCPUs if your database uses OCPUs). It is not possible to change the default ORDS configuration.
> See About Customer Managed Oracle REST Data Services on Autonomous Database for information on using an additional alternative ORDS deployment that enables flexible configuration options.

See Oracle REST Data Services for information on using Oracle REST Data Services.

See Predefined Database Service Names for Autonomous Database for information on the `low` database service.

Access RESTful Services and SODA for REST

Each Autonomous Database includes Oracle REST Data Services (ORDS) that provides HTTPS interfaces for working with the contents of your Oracle Database in REST enabled schemas.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To use Oracle REST Data Services and SODA for REST:

1. From the Autonomous Database details page select **Database Actions** and in the list click **View all database actions**.
2. On the Database Actions launchpad, under **Related Services**, click the **RESTful Services and SODA** card to see the base URL.
3. Click **Copy** to copy the URL.

If you are using Always Free Autonomous Database with Oracle Database 21c or with Oracle Database 23ai Oracle recommends the following:

- For projects that were started using a database release prior to Oracle Database 21c, explicitly specify the metadata for the default collection as specified in the example in the section SODA Drivers.
- For projects started using release Oracle Database 21c or later, just use the default metadata.

See SODA Drivers for more information.

Develop with Oracle REST Data Services on Autonomous Database

Autonomous Database supports Oracle REST Data Services (ORDS).

Developing RESTful services is easy with the following development interfaces:

- Database Actions (SQL Developer Web): Use Database Actions on Autonomous Database to REST enable users. See Manage Users and User Roles on Autonomous Database - Connecting with Database Actions for more information.
- SQL Developer (desktop): With SQL Developer on your desktop, you can connect to your database and enable REST services access to tables and views, or develop custom RESTful Services based on your SQL and PL/SQL code. See Connect Oracle SQL Developer with a Wallet (mTLS) for more information.
- Oracle APEX: With APEX you can use the RESTful Services development pages to build and maintain your services and REST enabled objects. You can use the APEX SQL Workshop to access your Oracle RESTful Services and REST enabled objects. See How to Access RESTful Services for more information.

The Autonomous Database `ADMIN` account is REST Enabled. This allows for REST Services to be published in the ADMIN schemas and allows you to access Database Actions using the ADMIN database user account. Oracle recommends you create an application schema account for your RESTful Services and REST enabled objects. Services are secured using Database Authentication and your REST enabled schema.

The authenticated database user is only permitted access if the schema is REST enabled and the URL mapping for the request points to their own schema. A user is not authenticated when a request points to any other database schema. For example, the following request authenticated as the REST enabled schema `HR` is accessible:

```
GET /ords/hr/module/service/
```

However, when authenticated as the REST enabled schema `SCOTT`, the same request:

```
GET /ords/hr/module/service/
```

results in an error:

```
401 HTTP Unauthorized response/error
```

Any database user whose credentials are correct and meets these rules is authenticated and granted the ORDS, mid-tier, role: `SQL Developer`. The `SQL Developer` role enables the user to access any endpoint that requires the `SQL Developer` role.

See REST-Enable a Database Table in *Quick Start Guide* for information on how to enable a table for REST access.

Use SODA for REST with Autonomous Database

Autonomous Database supports Simple Oracle Document Access (SODA) for REST.

- Overview of Using SODA for REST

- Load Purchase-Order Sample Data Using SODA for REST
  Oracle provides a substantial set of JSON purchase-order documents, in plain-text file `POList.json`, as a JSON array of objects, where each such object represents a document.
- Use SODA for REST with OAuth Client Credentials
  You can access SODA for REST on Autonomous Database using OAuth authentication. Depending on your application, accessing SODA for REST with OAuth authentication can improve performance and security.

Overview of Using SODA for REST

SODA for REST is a predeployed REST service that can be used to store JSON documents in your database.

SODA enables flexible, NoSQL-style application development without having to use SQL. With SODA, JSON documents are stored in named collections and managed using simple CRUD operations (create, read, update and delete). And while SQL isn't required, JSON stored in SODA collections is still fully accessible from SQL when needed. For example, an operational application may be fully built using SODA (without SQL) but then the data may be later analyzed using SQL from outside of the application. Autonomous Database SODA gives application developers the best of the NoSQL and SQL worlds - fast, flexible, and scalable application development without losing the ability to leverage SQL for analytics and reporting.

SODA for REST is deployed in ORDS under the following URL pattern, where *schema* corresponds to a REST-enabled database schema.

```
/ords/schema/soda/latest/*
```

The following examples use the cURL command line tool (http://curl.haxx.se/) to submit REST requests to the database. However, other 3rd party REST clients and libraries should work as well. The examples use database schema `ADMIN`, which is REST-enabled. You can SODA for REST with cURL commands from the Oracle Cloud Shell.

This command creates a new collection named "fruit" in the `ADMIN` schema:

```
> curl -X PUT -u 'ADMIN:<password>' \
"https://example-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/fruit"
```

These commands insert three JSON documents into the fruit collection:

```
> curl -X POST -u 'ADMIN:<password>' \
 -H "Content-Type: application/json" --data '{"name":"orange", "count":42}' \
 "https://example-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/fruit"

{"items":[{"id":"6F7E5C60197E4C8A83AC7D7654F2E375"...

> curl -X POST -u 'ADMIN:<password>' \
 -H "Content-Type: application/json" --data '{"name":"pear", "count":5}' \
 "https://example-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/fruit"

{"items":[{"id":"83714B1E2BBA41F7BA4FA93B109E1E85"...

> curl -X POST -u 'ADMIN:<password>' \
 -H "Content-Type: application/json" \
 --data '{"name":"apple", "count":12, "color":"red"}' \
 "https://example-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/fruit"
```

```
{"items":[{"id":"BAD7EFA9A2AB49359B8F5251F0B28549"...
```

This example retrieves a stored JSON document from the collection:

```
> curl -X POST -u 'ADMIN:<password>' \
 -H "Content-Type: application/json" --data '{"name":"orange"}' \
 "https://example-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/fruit?
action=query"

{
  "items": [
    {
      "id":"6F7E5C60197E4C8A83AC7D7654F2E375",
      "etag":"57215643953D7C858A7CB28E14BB48549178BE307D1247860AFAB2A958400E16",
      "lastModified":"2019-07-12T19:00:28.199666Z",
      "created":"2019-07-12T19:00:28.199666Z",
      "value":{"name":"orange", "count":42}
    }
  ],
  "hasMore":false,
  "count":1
}
```

This SQL query accesses the fruit collection:

```
SELECT
     f.json_document.name,
     f.json_document.count,
     f.json_document.color
FROM fruit f;
```

The query returns these three rows:

```
name      count     color
--------- --------- -------
orange    42        null
pear      5         null
apple     12        red
```

> **Note:**
>
> If you are using Always Free Autonomous Database with Oracle Database 21c, Oracle recommends the following:
> For projects that were started using a database release prior to Oracle Database 21c, explicitly specify the metadata for the default collection as specified in the example in the section SODA Drivers. For projects started using release Oracle Database 21c or later, just use the default metadata. See SODA Drivers for more information.

These examples show a subset of the SODA and SQL/JSON features. See the following for more information:

- SODA for REST for complete information on Simple Oracle Document Access (SODA)
- SODA for REST HTTP Operations for information on the SODA for REST HTTP operations

Load Purchase-Order Sample Data Using SODA for REST

Oracle provides a substantial set of JSON purchase-order documents, in plain-text file `POList.json`, as a JSON array of objects, where each such object represents a document.

The following examples use the cURL command line tool (http://curl.haxx.se/) to submit REST requests to the database. However, other 3rd party REST clients and libraries should work as well. The examples use database schema `ADMIN`, which is REST-enabled. You can use SODA for REST with cURL commands from the Oracle Cloud Shell.

You can load this sample purchase-order data set into a collection `purchaseorder` on your Autonomous Database with SODA for REST, using these curl commands:

```
curl -X GET "https://raw.githubusercontent.com/oracle/db-sample-schemas/
master/order_entry/POList.json" -o POList.json

curl -X PUT -u 'ADMIN:password' \
"https://example-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/
latest/purchaseorder"

curl -X POST -H -u 'ADMIN:password' 'Content-type: application/json' -d
@POList.json \
"https://example-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/
latest/purchaseorder?action=insert"
```

You can then use this purchase-order data to try out examples in *Oracle Database JSON Developer's Guide*.

For example, the following query selects both the `id` of a JSON document and values from the JSON purchase-order collection stored in column `json_document` of table `purchaseorder`. The values selected are from fields `PONumber`, `Reference`, and `Requestor` of JSON column `json_document`, which are projected from the document as virtual columns (see SQL NESTED Clause Instead of JSON_TABLE for more information).

```
SELECT id, t.*
  FROM purchaseorder
    NESTED json_document COLUMNS(PONumber, Reference, Requestor) t;
```

See the following for more information:

- SODA for REST for complete information on Simple Oracle Document Access (SODA)
- SODA for REST HTTP Operations for information on the SODA for REST HTTP operations

Use SODA for REST with OAuth Client Credentials

You can access SODA for REST on Autonomous Database using OAuth authentication. Depending on your application, accessing SODA for REST with OAuth authentication can improve performance and security.

Perform the following steps to use OAuth authentication to provide limited access to SODA for REST on Autonomous Database:

1. As the ADMIN user, access Database Actions and create a user with the required privileges.
   a. Access Database Actions as ADMIN.

      See Access Database Actions as ADMIN for more information.

   b. In Database Actions, click ☰ to show the available actions.
   c. In Database Actions, under **Administration** select **Database Users**.
   d. Click **Create User**.
   e. In the **Create User** area, on the **User** tab enter **User Name** and a **Password** and confirm the password.
   f. Select **Web Access**.
   g. In the **Create User** area, select the **Granted Roles** tab and grant `DWROLE` to the user.
   h. Click **Create User**.

   See Manage Users and User Roles on Autonomous Database - Connecting with Database Actions for more information.
2. Use a SQL worksheet in Database Actions to grant user privileges required to load data.
   a. Access Database Actions as ADMIN.

      See Access Database Actions as ADMIN for more information.

   b. In Database Actions, click ☰ to show the available actions.
   c. In Database Actions, under **Development** click **SQL** to open a SQL worksheet.
   d. Grant user privileges required to load data to the user from Step 1.

      ```
      GRANT UNLIMITED TABLESPACE TO user_name;
      ```

   See Manage User Privileges on Autonomous Database - Connecting with a Client Tool for more information.
3. Sign out as the ADMIN user.
4. Sign in to Database Actions as the user that is setting up to use OAuth authentication.
5. In Database Actions, use a SQL worksheet to register the OAuth client.
   a. Register the OAuth client.

      For example, enter the following commands into the SQL worksheet, where you supply the appropriate values for your user and your client application.

      ```
      BEGIN
        OAUTH.create_client(
          p_name            => 'my_client',
      ```

```
    p_grant_type      => 'client_credentials',
    p_owner           => 'Example Company',
    p_description     => 'A client for my SODA REST resources',
    p_support_email   => 'user_name@example.com',
    p_privilege_names => 'my_priv'
  );

  OAUTH.grant_client_role(
    p_client_name => 'my_client',
    p_role_name   => 'SQL Developer'
  );

  OAUTH.grant_client_role(
    p_client_name => 'my_client',
    p_role_name   => 'SODA Developer'
  );
  COMMIT;
END;
/
```

b. In the SQL worksheet, click **Run Script** to run the command.

See OAUTH PL/SQL Package Reference for more information.

This registers a client named `my_client` to access the `my_priv` privilege using OAuth client credentials.

6. Obtain the `client_id` and `client_secret` required to generate the access token.

For example, in the SQL worksheet run the following command:

```
SELECT id, name, client_id, client_secret FROM user_ords_clients;
```

7. Obtain the access token. To get an access token you send a `REST GET` request to *`database_ORDS_urluser_name`*`/oauth/token`.

The *`database_ORDS_url`* is available from Database Actions, under **Related Services**, on the **RESTful Services and Soda** card. See Access RESTful Services and SODA for REST for more information.

In the following command, use the `client_id` and the `client_secret` you obtained in Step 6.

The following example uses the `cURL` command line tool (http://curl.haxx.se/) to submit REST requests to Autonomous Database. However, other 3rd party REST clients and libraries should work as well.
You can use the `cURL` command line tool to submit the `REST GET` request. For example:

```
> curl -i -k --user SBA-iO9Xe12cdZHYfryBGQ..:vvUQ1AagTqAqdA2oN7afSg.. --
data "grant_type=client_credentials"https://mqssyowmqvgac1y-
doc.adb.region.oraclecloudapps.com/ords/user_name/oauth/token
HTTP/1.1 200 OK
Date: Mon, 22 Jun 2020 15:17:11 GMT
Content-Type: application/jsonTransfer-Encoding: chunked
Connection: keep-alive
X-Frame-Options: SAMEORIGIN
```

```
{"access_token":"JbOKtAuDgEh2DXx0QhvPGg","token_type":"bearer","expires_in"
:3600}
```

To specify both the `client_id` and the `client_secret` with the curl `--user` argument, enter a colon to separate the `client_id` and the `client_secret`. If you only specify the user name, `client_id`, curl prompts for a password and you can enter the `client_secret` at the prompt.

8. Use the access token to access the protected resource.

   The token obtained in the previous step is passed in the Authorization header. For example:

```
> curl -i -H "Authorization: Bearer JbOKtAuDgEh2DXx0QhvPGg" -X GET https://
database_id.adb.region.oraclecloudapps.com/ords/user_name/soda/latest
HTTP/1.1 200 OK
Date: Mon, 22 Jun 2020 15:20:58 GMT
Content-Type: application/json
Content-Length: 28
Connection: keep-alive
X-Frame-Options: SAMEORIGIN
Cache-Control: private,must-revalidate,max-age=0


{"items":[],"hasMore":false}
```

See Configuring Secure Access to RESTful Services for complete information on secure access to RESTful Services.

Access Oracle REST Data Services, Oracle APEX, and Developer Tools Using a Vanity URL

By default you access Oracle REST Data Services (ORDS) endpoints, Oracle APEX apps, and developer tools on Autonomous Database using the `oraclecloudapps.com` domain name.

To access Oracle REST Data Services (ORDS) endpoints, you can optionally configure a vanity URL or custom domain name that is easy to remember to help promote your brand identity.

After you acquire a desired domain name and matching SSL certificate from a vendor of your choice, deploy an Oracle Cloud Infrastructure Load Balancer in your Virtual Cloud Network (VCN) using your Autonomous Database as the backend. Your Autonomous Database instance must be configured with a private endpoint in the same VCN. See Configuring Network Access with Private Endpoints for more information.

To learn more, see the following:

- Introducing Vanity URLs for APEX and ORDS on Oracle Autonomous Database
- Automate Vanity URL Configuration Using Terraform

About Customer Managed Oracle REST Data Services on Autonomous Database

Optionally, you can configure Autonomous Database to use Oracle REST Data Services (ORDS) running in a customer managed environment.

When you provision an Autonomous Database instance, by default Oracle REST Data Services (ORDS) is preconfigured and available for the instance. With the default Oracle REST Data Services, Oracle performs any required configuration, patching, and maintenance. When you use the default ORDS on Autonomous Database, you cannot modify any of the ORDS configuration options.

Use a customer managed environment if you want manual control of the configuration and management of Oracle REST Data Services. For example, use this option when your applications require larger connection pools or if you need more control over the ORDS configuration options.

When ORDS runs in a customer managed environment, you are responsible for configuration, patching, and maintenance of ORDS. After you configure Autonomous Database to use your customer managed ORDS in addition to the existing autonomously managed ORDS, you can route ORDS HTTPS traffic through your environment. The default Autonomous Database web server and ORDS are still running and ORDS traffic goes to the ORDS running in the customer managed environment. This provides an additional and alternative HTTPS solution for Autonomous Database.

Installing and configuring a customer managed environment for ORDS allows you to run ORDS with configuration options that are not possible using the default Oracle managed ORDS available with Autonomous Database.

See Installing and Configuring Customer Managed ORDS on Autonomous Database for more information.

Installing and configuring a customer managed environment for ORDS is only supported with Autonomous Database Serverless. Your Autonomous Database must be configured for one of the following workload types: Data Warehouse, Transaction Processing, or JSON Database.

> **Note:**
>
> Oracle REST Data Services 19.4.6 or higher is required to use a customer managed environment for ORDS with Autonomous Database. Customer managed environment for ORDS is not supported if your database is configured for APEX workload type.

Using Oracle Database API for MongoDB

Oracle Database API for MongoDB makes it possible to connect to Oracle Autonomous Database using MongoDB language drivers and tools.

Oracle Database API for MongoDB leverages the converged database capabilities of an Autonomous Database to manage multiple data types, including JSON data, within a single database. For example, these converged database capabilities allow you to use SQL to query or update JSON data.

See Oracle Database API for MongoDB for more information.

See About Autonomous JSON Database for more information.

See About Autonomous Database Workload Types for more information.

- Configure Access for MongoDB and Enable MongoDB

- User Management for MongoDB
  Oracle Database API for MongoDB enables you to use an Oracle Autonomous Database as the data store. If you want or need to use an existing Autonomous Database for this purpose, here is the workflow.
- Create a Test Autonomous Database User for MongoDB
- Connect MongoDB Applications to Autonomous Database
  Connecting your MongoDB application to Autonomous Database includes several steps, depending upon your requirements.

Configure Access for MongoDB and Enable MongoDB

Oracle Database API for MongoDB enables you to use an Oracle Autonomous Database as the data store.

To use the MongoDB API you can create and configure a new Autonomous Database or modify the configuration of an existing Autonomous Database. MongoDB requires that you configure network access to use ACLs or that you define a private endpoint for the Autonomous Database instance. In addition to configuring the network access, you must enable MongoDB API on the Autonomous Database instance.

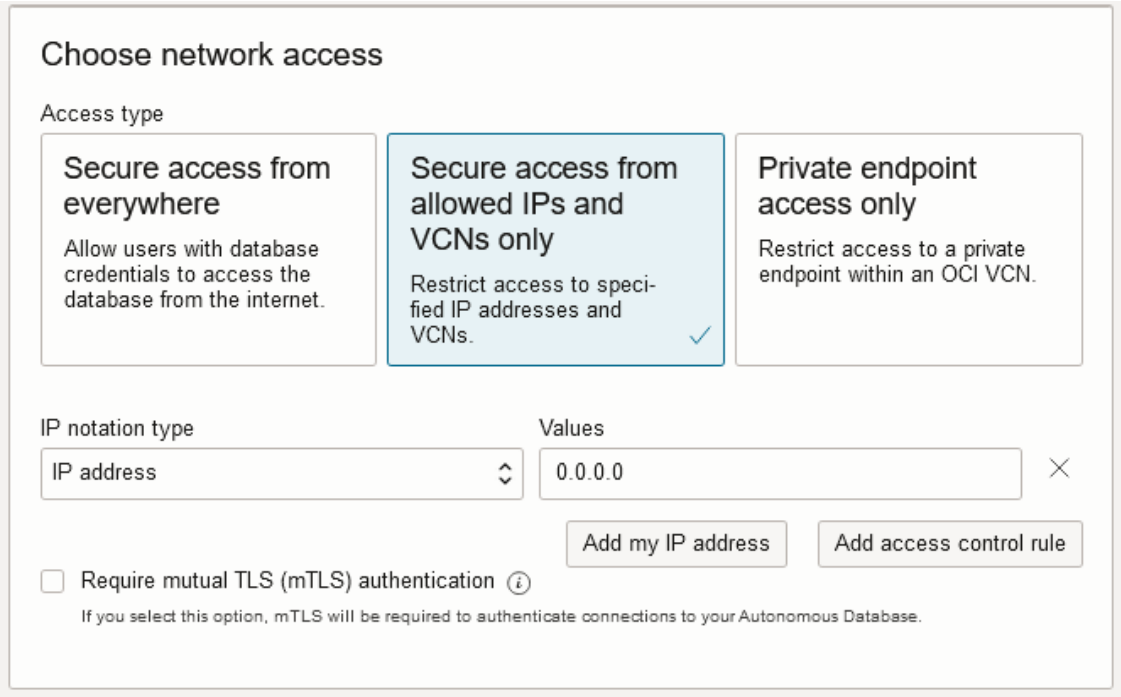
- Configure Access for MongoDB
  To use the MongoDB API, you can create and configure a new Autonomous Database or modify the configuration of an existing Autonomous Database by configuring ACLs or by defining a private endpoint.
- Enable MongoDB API on Autonomous Database
  After you configure the network access for the Autonomous Database instance, enable the MongDB API.

Configure Access for MongoDB

To use the MongoDB API, you can create and configure a new Autonomous Database or modify the configuration of an existing Autonomous Database by configuring ACLs or by defining a private endpoint.

Configure a New Autonomous Database for MongoDB

Follow the steps in Provision or Clone an Autonomous Database, up to the point where you select your Network Access Type.



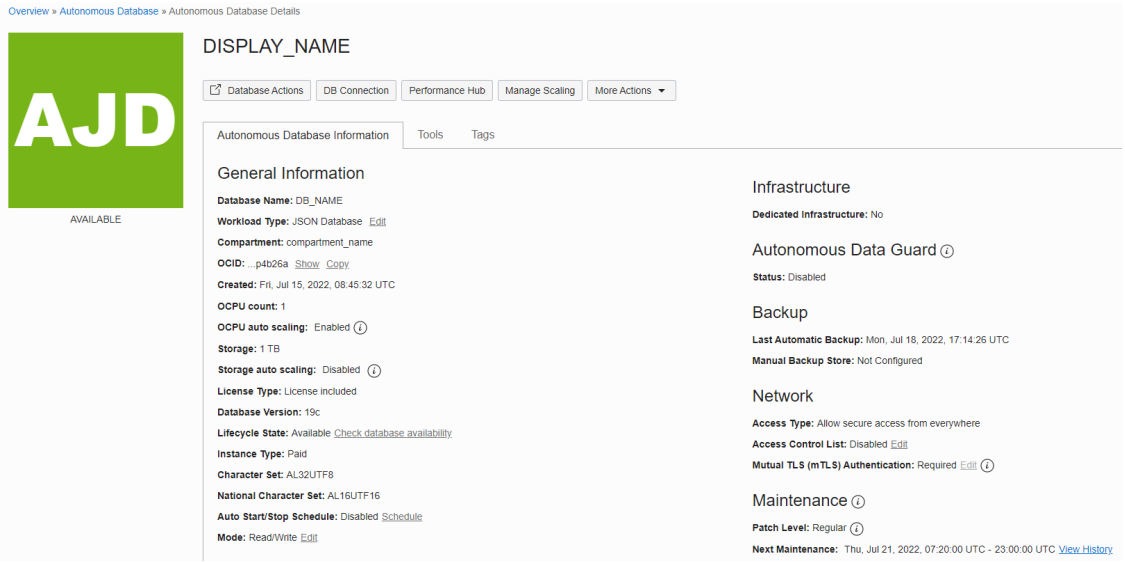
At this point, to use Oracle Database API for MongoDB configure secure access by selecting and configuring one of these network access types:

- **Secure access from allowed IPs and VCNs only**
- **Private endpoint access only**

  See Configure Network Access with Private Endpoints for information on configuring an Autonomous Database instance with a private endpoint.

Configure an Existing Autonomous Database for MongoDB

Open the Oracle Cloud Infrastructure Console for your Autonomous Database instance.



> **Note:**
>
> To use Oracle Database API for MongoDB the Network must be configured and the **Access type** must be either: **Allow secure access from specified IPs and VCNs** or **Virtual Cloud Network**.

Access Control List (ACL) Setup

See Configure Access Control Lists for an Existing Autonomous Database Instance for more information.

To configure your ACL for an IP address, you will need to obtain the public IP address. There are several ways to show your public IP address:

- In the choose network access area, click **Add My IP Address**. This copies your IP address into the **Values** field.
- After disabling any VPN, use the WhatIsMyIP website.
- After disabling any VPN, use the curl command: `curl -s https://ifconfig.me`.

> **Note:**
>
> Public IP addresses may change. Any change to your public IP address will require a change in the ACL. If you are unable to access your database, your ACL should be something you check.

ACLs Types and Use Cases

ACL Type	Use Case	Comment
IP Address	Local development laptops sharing the same public IP address	Easiest way to get started. Any laptop connected on this LAN will have access to the database with the database credentials.
CIDR Block	Local development laptop	Using IPv4/32 notation
IP Addresses separated by commas	Small number of local development laptops connected on distinct LANs (having distinct public IP addresses)	Can be tedious to manage with 10+ laptops.
CIDR Block	Local development laptops connected on the same subnet exposed to Internet (each laptop has its own public IP Address)	Rely on CIDR Block notation. See calculator here for more information. Example: `89.84.109.0/24` gives 256 possible IP addresses from `89.84.109.0` to `89.84.109.255`
VCN with CIDR Block	For testing, production, or CI/CD pipeline hosted on OCI having their own VCN and Compute instances	Assign OCI compartment per environment type.

ACL Type	Use Case	Comment
Mixing IP Address and VCN with CIDR Block	Local development laptop accessing a test Autonomous Database with connections from the testing environment or CI/CD pipeline	A common configuration option for on-going development work.

Enable MongoDB API on Autonomous Database

After you configure the network access for the Autonomous Database instance, enable the MongDB API.

To enable the MongoDB API for an existing instance:

1. On the Autonomous Database details page, select the **Tool configuration** tab.
2. Click **Edit tool configuration**.
3. In the MongoDB API row, select in the **Enable tool** column to show **Enabled**.
4. Click **Apply**.

The **Lifecycle state** changes to updating until MongoDB is enabled.

You can also enable the MongoDB API when you provision or clone an instance by selecting **Show advanced options** and selecting the **Tools** tab.

See Manage Autonomous Database Built-in Tools for more information.

User Management for MongoDB

Oracle Database API for MongoDB enables you to use an Oracle Autonomous Database as the data store. If you want or need to use an existing Autonomous Database for this purpose, here is the workflow.

Oracle Database API for MongoDB enables the mapping of Autonomous Database objects to MongoDB objects as follows:

MongoDB Object	Oracle Autonomous Database Object
database	schema
collection	table
document	document (in a column)

For example, you could create a collection using the Oracle Database API for MongoDB as follows:

```
use scott;
db.createCollection('fruit');
```

A table named **FRUIT** is created in the schema **SCOTT**.

When you connect to the Oracle Database API for MongoDB, you authenticate using an Autonomous Database username and password. This authenticated connection then accesses collections within the corresponding schema. This user must meet the following requirements:

- The user's schema must be ORDS-enabled, which is sometimes referred to as enabled for Web Access. See Basic Setup to Enable ORDS Database API for more information.

- The user must have the following roles and privileges: `SODA_APP`, `CREATE TABLE`, and `CREATE SESSION`. See Manage User Roles and Privileges on Autonomous Database for more information.
- The user has a quota on tablespace **DATA**. See Create Users on Autonomous Database for more information.

> **Note:**
>
> The role `DWROLE` in the Autonomous Database contains these roles, among others.

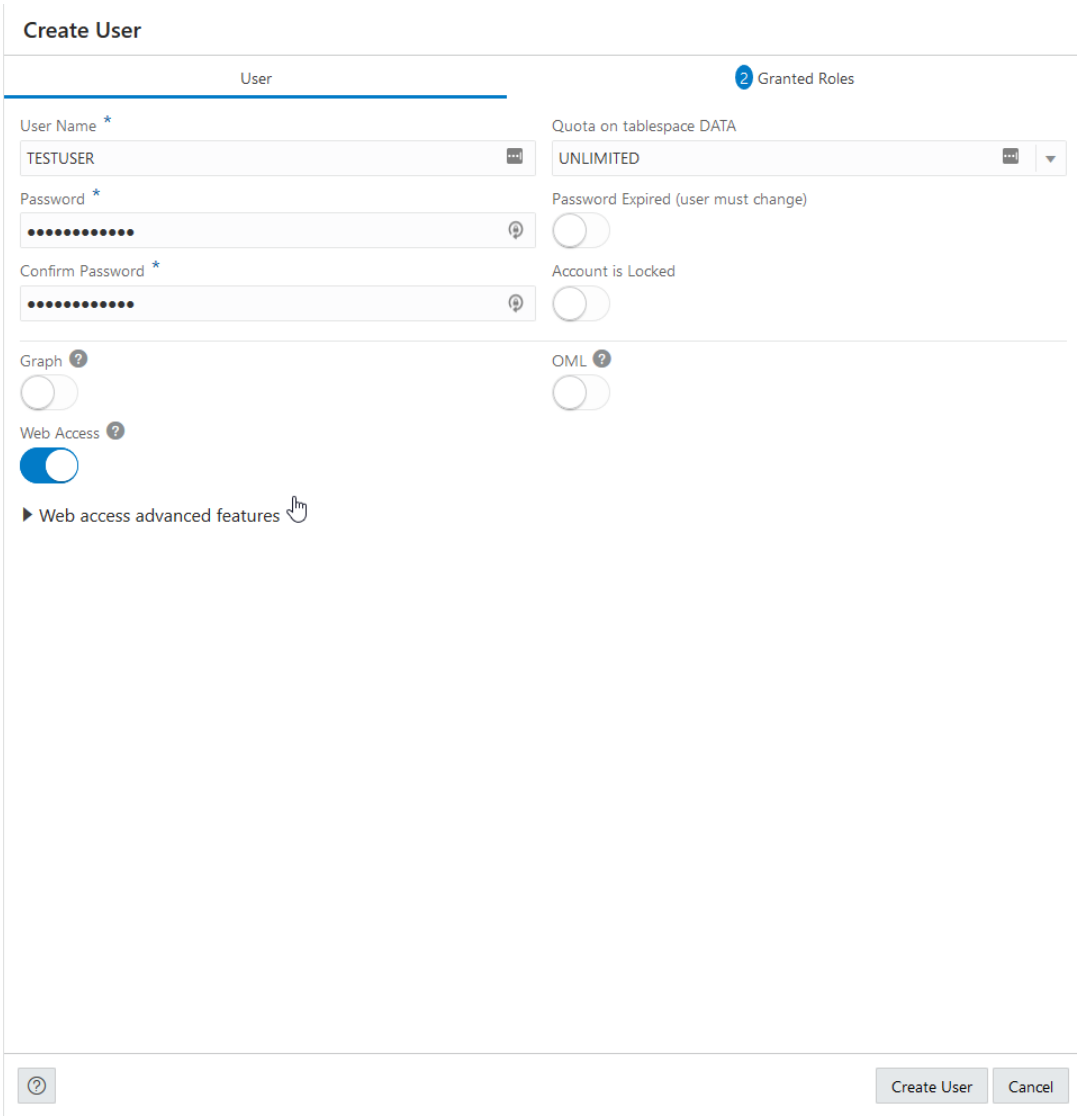
Access to schemas not granted to the user is prohibited. For example, the user **SCOTT** can only access collections in the schema **SCOTT**. There is one exception. If the authenticated user has the Autonomous Database privileges `CREATE USER`, `ALTER USER` and `DROP USER`, that user can access any ORDS-enabled schema.

Additionally, a user with these privileges can implicitly create schemas. That is, when the user creates a collection in a database that does not exist, the schema will be created automatically. See Oracle Database API for MongoDB for more information.

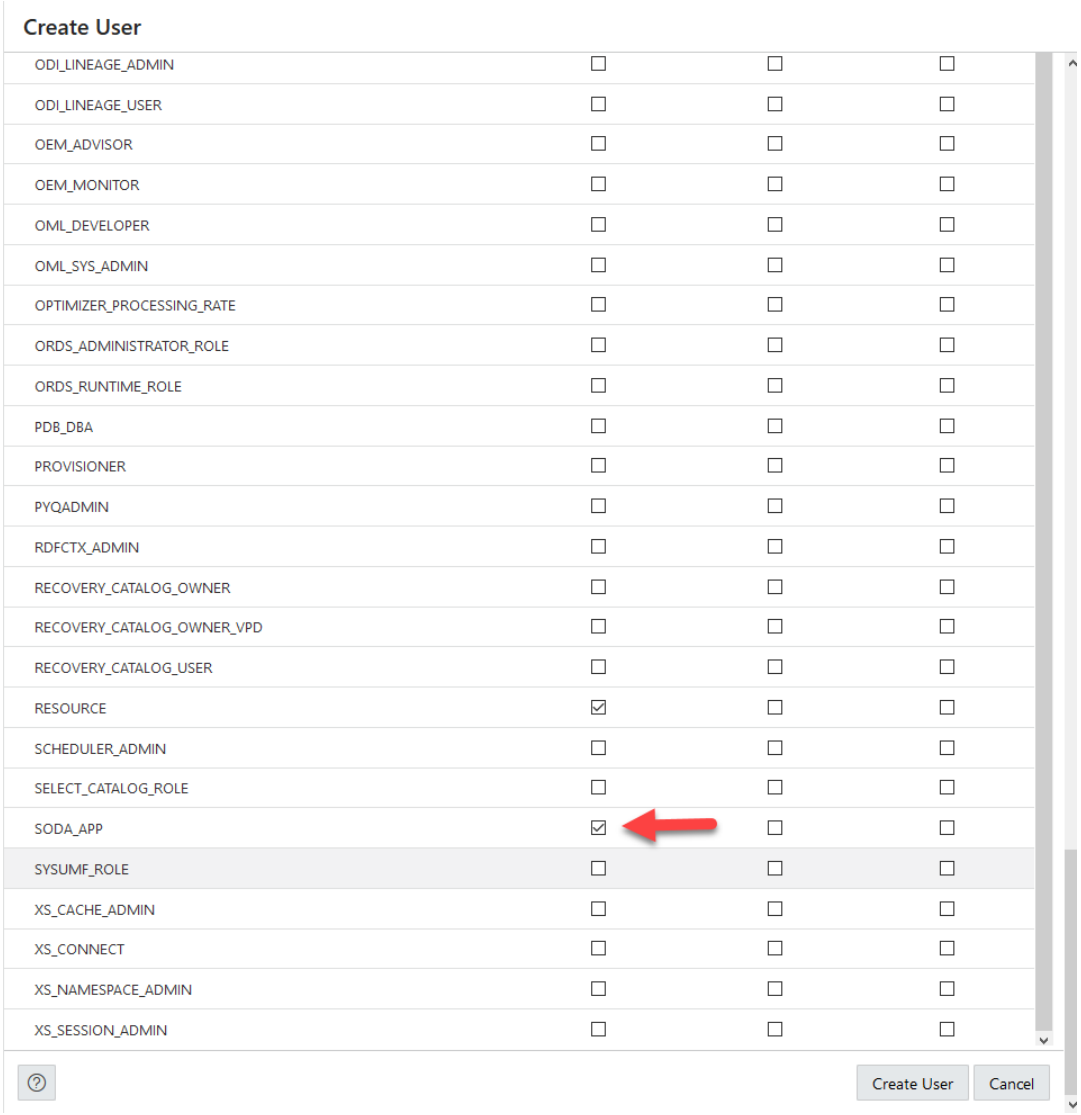
Create a Test Autonomous Database User for MongoDB

The steps that follow use Database Actions to create a test user with the proper roles. You can also use Database Actions SQL or other SQL command line utilities to execute the SQL statements directly. See Create Users on Autonomous Database - Connecting with a Client Tool for more information.

1. Open the Oracle Cloud Infrastructure Console for your Autonomous Database.
2. Select **Database Actions**.
3. From the Database Actions Launchpad, select **Administration** > **Database Users**.
4. Select **+ Create User** on the Database Users page.
5. Enter a **User Name** and **Password** for your test user. Select **Web Access**, and set the **Quota on tablespace DATA**.



6. Select the **Granted Roles** tab.
7. In addition to the default roles, select and add the `SODA_APP` role for the user.



8. Select the **Create User** button.

You can user this user, or a similarly authenticated user, for testing purposes. See Test Connection Using the Command Line for more information.

Connect MongoDB Applications to Autonomous Database

Connecting your MongoDB application to Autonomous Database includes several steps, depending upon your requirements.

Topics

- Retrieve the Autonomous Database Connection String from Database Actions
- Test Connection Using the Command Line
- Test Connection Using a Node.js Application
- Retrieve the Autonomous Database MongoDB Connection String
- Test Connection Using the Command Line
- Test Connection Using a Node.js Application

Retrieve the Autonomous Database MongoDB Connection String

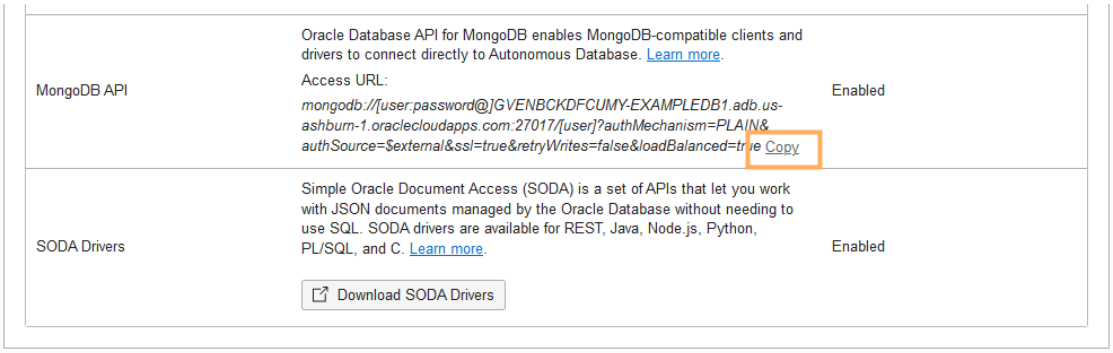
You can retrieve the MongoDB API connection string from the Oracle Cloud Infrastructure Console.

After you obtain the MongoDB API connection string you can use the MongoDB Shell, which is a command-line utility, to connect and query your data.

First you must configure network access and enable the MongoDB API. See Configure Access for MongoDB for more information.

To retrieve the MongoDB API connection string:

1. On the Autonomous Database details page, select the **Tool configuration** tab.
2. In the MongoDB API row, under Access URL click **Copy**.



- Retrieve the Autonomous Database Connection String from Database Actions

Retrieve the Autonomous Database Connection String from Database Actions

Use the MongoDB Shell which is a command-line utility used to connect and query your data.

1. You can retrieve the connection string for your Autonomous Database instance with Database Actions.

   See Access Database Actions as ADMIN for more information.
2. On the Database Actions Launchpad, under **Related Services**, click **Oracle Database API for MongoDB**.
3. On the **Oracle Database API for MongoDB** page click **Copy**.

Test Connection Using the Command Line

1. Login as the test user. See Create a Test Autonomous Database User for MongoDB for more information.

```
$ mongosh --tls --tlsAllowInvalidCertificates 'mongodb://
TESTUSER:<PASSWORD>@<database URL>.
<OCI region>.oraclecloudapps.com:27017/admin?
authMechanism=PLAIN&authSource=$external&ssl=true&loadBalanced=false'
Current Mongosh Log ID: 614c9e2a01e3575c8c0b2ec7
Connecting to:          mongodb://TESTUSER:<PASSWORD>@<database
URL>.<OCIregion>.oraclecloudapps.com:27017/admin?
authMechanism=PLAIN&authSource=$external&tls=true&loadBalanced=false
```

```
Using MongoDB:                  3.6.2
Using Mongosh:                  1.0.7
For mongosh info see: https://docs.mongodb.com/mongodb-shell/admin
> show dbs
testuser    0 B
>
```

Note:

Use URI percent-encoding to replace any reserved characters in your connection-string URI — in particular, characters in your username and password. These are the reserved characters and their percent encodings:

!	#	$	%	&	'	(	)	*	+
%21	%23	%24	%25	%26	%27	%28	%29	%2A	%2B

,	/	:	;	=	?	@	[	]
%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

For example, if your username is `RUTH` and your password is `@least1/2#?` then your MongoDB connection string to server *<server>* might look like this:

```
'mongodb://RUTH:%40least1%2F2%23%3F@<server>:27017/ruth/ ...'
```

Depending on the tools or drivers you use, you might be able to provide a username and password as separate parameters, instead of as part of a URI connection string. In that case you likely won't need to encode any reserved characters they contain.

See also:

- Percent Encoding - Reserved Characters
- Uniform Resource Identifier (URI): Generic Syntax

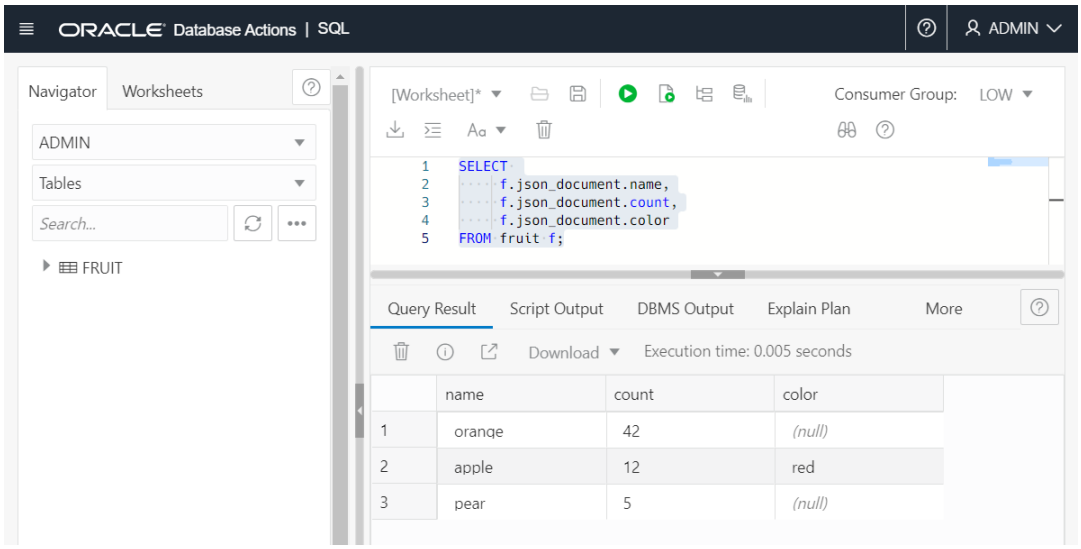
2. Create a collection and insert documents into your collection.

```
testuser> show collections

testuser> db.createCollection( 'fruit' )
{ ok: 1 }
testuser> show collections
fruit
testuser> db.fruit.insertOne( {name:"orange", count:42} )
{
  acknowledged: true,
  insertedId: ObjectId("614ca31fdab254f63e4c6b47")
}
testuser> db.fruit.insertOne( {name:"apple", count:12, color: "red"} )
{
  acknowledged: true,
  insertedId: ObjectId("614ca340dab254f63e4c6b48")
```

```
}
testuser> db.fruit.insertOne( {name:"pear", count:5} )
{
  acknowledged: true,
  insertedId: ObjectId("614ca351dab254f63e4c6b49")
}
testuser>
```

3. Query the collection using a SQL client such as Database Actions.

```
SELECT
     f.json_document.name,
     f.json_document.count,
     f.json_document.color
FROM fruit f;
```



Test Connection Using a Node.js Application

1. Download Node.js. If you have already downloaded or installed Node.js for your environment, you can skip this step.

```
$ wget https://nodejs.org/dist/latest-v14.x/node-v14.17.5-linux-x64.tar.xz
```

2. Extract the contents of the Node,js archive. If you have already downloaded or installed Node.js for your environment, you can skip this step.

```
$ tar -xf node-v14.17.5-linux-x64.tar.xz
```

3. Configure the `PATH` environment variable. If you have already downloaded or installed Node.js for your environment, you can skip this step.

```
$ export PATH="`pwd`"/node-v14.17.5-linux-x64/bin:$PATH
```

4. Test your connection with a Javascript example.

a. Create a new directory.

```
$ mkdir autonomous_mongodb
$ cd autonomous_mongodb
$ npm init -y
```

b. Install mongodb dependency.

```
$ npm install mongodb
```

c. Create a JavaScript application named `connect.js`.

```
const { MongoClient } = require("mongodb");
const uri =
   "mongodb://TESTUSER:<PASSWORD>@<Database URI>.<OCI
region>.oraclecloudapps.com:27017/admin?
authMechanism=PLAIN&authSource=$external&ssl=true&loadBalanced=false";

const client = new MongoClient(uri);

async function run() {
  try {
        await client.connect();

        const database = client.db('admin');
        const movies = database.collection('movies');

        // Insert a movie
        const doc = { title: 'Back to the Future',
                      year: 1985, genres: ['Adventure', 'Comedy', 'Sci-
Fi'] }
        const result = await movies.insertOne(doc);

        // Query for a movie that has the title 'Back to the Future' :)
        const query = { title: 'Back to the Future' };
        const movie = await movies.findOne(query);

        console.log(movie);
  } finally {
     // Ensures that the client will close when you finish/error
     await client.close();
  }
}
run().catch(console.dir);
```

> **Note:**
>
> Use URI percent-encoding to replace any reserved characters in your connection-string URI — in particular, characters in your username and password. These are the reserved characters and their percent encodings:
>
> | ! | # | $ | % | & | ' | (|) | * | + |
> |---|---|---|---|---|---|---|---|---|---|
> | %21 | %23 | %24 | %25 | %26 | %27 | %28 | %29 | %2A | %2B |
>
> | , | / | : | ; | = | ? | @ | [|] |
> |---|---|---|---|---|---|---|---|---|
> | %2C | %2F | %3A | %3B | %3D | %3F | %40 | %5B | %5D |
>
> For example, if your username is `RUTH` and your password is `@least1/2#?` then your MongoDB connection string to server *<server>* might look like this:
>
> `'mongodb://RUTH:%40least1%2F2%23%3F@<server>:27017/ruth/ ...'`
>
> Depending on the tools or drivers you use, you might be able to provide a username and password as separate parameters, instead of as part of a URI connection string. In that case you likely won't need to encode any reserved characters they contain.
>
> See also:
>
> - Percent Encoding - Reserved Characters
> - Uniform Resource Identifier (URI): Generic Syntax

d. Run the example. The output should look similar to the following.

```
$ node connect
{
  _id: new ObjectId("611e3266005202371acf27c1"),
  title: 'Back to the Future',
  year: 1985,
  genres: [ 'Adventure', 'Comedy', 'Sci-Fi' ]
}
```

Send Email and Notifications on Autonomous Database

There are a number of options for sending email on Autonomous Database. You can also send notifications to a Slack or MSTeams channel.

- Send Email on Autonomous Database
  There are a number of options for sending email on Autonomous Database. You can also send text messages or the output of an SQL query to a Slack or MSTeams channel.
- Send Slack Notifications from Autonomous Database
  Describes how to configure Slack so that you can send messages, alerts, or output of a query from Autonomous Database to a Slack Channel. Also describes the procedures you use to send Slack notifications.

- Send Microsoft Teams Notifications from Autonomous Database
  Describes how to configure Microsoft Teams so that you can send messages, alerts, or output of a query from Autonomous Database to a Microsoft Teams Channel. Also describes the procedures you use to send Microsoft Teams notifications.

Send Email on Autonomous Database

There are a number of options for sending email on Autonomous Database. You can also send text messages or the output of an SQL query to a Slack or MSTeams channel.

- Send Email with Email Delivery Service on Autonomous Database
  Describes the steps to send email using `UTL_SMTP` on Autonomous Database.
- Send Email with an Email Provider on a Private Endpoint
  Describes the steps to send email with an email provider that is on Private Endpoint.
- Use Credential Objects to set SMTP Authentication
  Describes how to pass a credential objects to `UTL_SMTP.SET_CREDENTIAL` APIs.
- Send Email from Autonomous Database Using DBMS_CLOUD_NOTIFICATION
  Use the `DBMS_CLOUD_NOTIFICATION` package to send messages and query results as email.

Send Email with Email Delivery Service on Autonomous Database

Describes the steps to send email using `UTL_SMTP` on Autonomous Database.

> **Note:**
>
> Oracle Cloud Infrastructure Email Delivery Service is the only supported email provider for public SMTP endpoints.

To send email with Oracle Cloud Infrastructure Email Delivery Service:

1. Identify your SMTP connection endpoint for Email Delivery. You may need to subscribe to additional Oracle Cloud Infrastructure regions if Email Delivery is not available in your current region.

   For example, select one of the following for the SMTP connection endpoint:

   - smtp.us-phoenix-1.oraclecloud.com
   - smtp.us-ashburn-1.oraclecloud.com
   - smtp.email.uk-london-1.oci.oraclecloud.com
   - smtp.email.eu-frankfurt-1.oci.oraclecloud.com

   See Configure SMTP Connection for more information.
2. Generate SMTP credentials for Email Delivery. `UTL_SMTP` uses credentials to authenticate with Email Delivery servers when you send email.

   See Generate SMTP Credentials for a User for more information.
3. Create an approved sender for Email Delivery. Complete this step for all email addresses you use as the "`From`" with `UTL_SMTP.MAIL`.

   See Managing Approved Senders for more information.
4. Allow SMTP access for ADMIN user by appending an Access Control Entry (ACE).

For example:

```
BEGIN
  -- Allow SMTP access for user ADMIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'www.us.example.com',
    lower_port => 587,
    upper_port => 587,
    ace => xs$ace_type(privilege_list => xs$name_list('SMTP'),
                       principal_name => 'ADMIN',
                       principal_type => xs_acl.ptype_db));
END;
/
```

5. Create a PL/SQL procedure to send email.

   For example, see the sample code shown in SMTP Send Email Sample Code.

6. Send a test email using the PL/SQL procedure you created in step 5.

   For example:

```
execute send_mail('taylor@example.com', 'Email from Oracle Autonomous
Database', 'Sent using UTL_SMTP');
```

See UTL_SMTP for information on `UTL_SMTP`.

See PL/SQL Packages Notes for Autonomous Database for `UTL_SMTP` restrictions with Autonomous Database.

- SMTP Send Email Sample Code
  Shows sample code for sending email with `UTL_SMTP` on Autonomous Database.

SMTP Send Email Sample Code

Shows sample code for sending email with `UTL_SMTP` on Autonomous Database.

```
CREATE OR REPLACE PROCEDURE SEND_MAIL (
  msg_to varchar2,
  msg_subject varchar2,
  msg_text varchar2 )
IS

  mail_conn utl_smtp.connection;
  username varchar2(1000):= 'ocid1.user.oc1.username';
  passwd varchar2(50):= 'password';
  msg_from varchar2(50) := 'adam@example.com';
  mailhost VARCHAR2(50) := 'smtp.us-ashburn-1.oraclecloud.com';

BEGIN
  mail_conn := UTL_smtp.open_connection(mailhost, 587);
  utl_smtp.starttls(mail_conn);

  UTL_SMTP.AUTH(mail_conn, username, passwd, schemes => 'PLAIN');

  utl_smtp.mail(mail_conn, msg_from);
  utl_smtp.rcpt(mail_conn, msg_to);
```

```
  UTL_smtp.open_data(mail_conn);

  UTL_SMTP.write_data(mail_conn, 'Date: ' || TO_CHAR(SYSDATE, 'DD-MON-YYYY
HH24:MI:SS') || UTL_TCP.crlf);
  UTL_SMTP.write_data(mail_conn, 'To: ' || msg_to || UTL_TCP.crlf);
  UTL_SMTP.write_data(mail_conn, 'From: ' || msg_from || UTL_TCP.crlf);
  UTL_SMTP.write_data(mail_conn, 'Subject: ' || msg_subject || UTL_TCP.crlf);
  UTL_SMTP.write_data(mail_conn, 'Reply-To: ' || msg_to || UTL_TCP.crlf ||
UTL_TCP.crlf);
  UTL_SMTP.write_data(mail_conn, msg_text || UTL_TCP.crlf || UTL_TCP.crlf);

  UTL_smtp.close_data(mail_conn);
  UTL_smtp.quit(mail_conn);

EXCEPTION
  WHEN UTL_smtp.transient_error OR UTL_smtp.permanent_error THEN
    UTL_smtp.quit(mail_conn);
    dbms_output.put_line(sqlerrm);
  WHEN OTHERS THEN
    UTL_smtp.quit(mail_conn);
    dbms_output.put_line(sqlerrm);
END;
/
```

Where:

- *mailhost*: specifies the SMTP Connection Endpoint from Step 1 in Send Email with Email Delivery Service on Autonomous Database.
- *username*: specifies the SMTP credential username from Step 2 in Send Email with Email Delivery Service on Autonomous Database.
- *passwd*: specifies the SMTP credential password from Step 2 in Send Email with Email Delivery Service on Autonomous Database.
- *msg_from*: specifies one of the approved senders from Step 3 in Send Email with Email Delivery Service on Autonomous Database.

Send Email with an Email Provider on a Private Endpoint

Describes the steps to send email with an email provider that is on Private Endpoint.

To send email from Autonomous Database using a email provider on a private endpoint, the email provider must be accessible from the Oracle Cloud Infrastructure VCN (the Autonomous Database instance's private endpoint). For example, you can access an email provider when:

- Both the source Autonomous Database instance and the email provider are in the same Oracle Cloud Infrastructure VCN.
- The source Autonomous Database instance and the email provider are in different Oracle Cloud Infrastructure VCNs that are paired.
- The email provider is on an on-premises network that is connected to the source Autonomous Database instance's Oracle Cloud Infrastructure VCN using FastConnect or VPN.

As a prerequisite, to send email using an email provider, define the following ingress and egress rules:

- Define an egress rule in the source database's subnet security list or network security group such that the traffic to the target host is allowed on port 587 or port 25 (depending on which port you are using).
- Define an ingress rule in the target host's subnet security list or network security group such that the traffic from the source Autonomous Database instance's IP address to port 587 or port 25 is allowed (depending on which port you are using).

> **Note:**
>
> Making calls to a private email provider is only supported in commercial regions and US Government regions.
> This feature is enabled by default in all commercial regions.
>
> This feature is enabled by default in US Government regions for newly provisioned databases.
>
> For existing US Government databases on a private endpoint, if you want to call an email provider from an Autonomous Database to a target in a US Government region, please file a Service Request at Oracle Cloud Support and request to enable the private endpoint in government regions database linking feature.
>
> US Government regions include the following:
>
> - Oracle Cloud Infrastructure US Government Cloud with FedRAMP Authorization
> - Oracle Cloud Infrastructure US Federal Cloud with DISA Impact Level 5 Authorization

To send email from an email provider on private endpoint:

1. Allow SMTP access for ADMIN user by appending an Access Control Entry (ACE).

   For example:

```
-- Create an Access Control List for the host
BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'www.example.com',
         lower_port => 587,
         upper_port => 587,
         ace => xs$ace_type(privilege_list => xs$name_list('SMTP'),
                            principal_name => 'ADMIN',
                            principal_type => xs_acl.ptype_db),
         private_target => TRUE);
END;
/
```

> **Note:**
>
> `DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE` only supports a single hostname for the `host` parameter (on a private endpoint, using an IP address, a SCAN IP, or a SCAN hostname is not supported).

If you set `ROUTE_OUTBOUND_CONNECTIONS` to `PRIVATE_ENDPOINT`, setting the `private_target` parameter to `TRUE` is not required in this API. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

2. Create a PL/SQL procedure to send email.
3. Send a test email using the PL/SQL procedure you created in step 2.

   For example:

```
execute send_mail('taylor@example.com', 'Email from Oracle Autonomous
Database', 'Sent using private email provider');
```

Use Credential Objects to set SMTP Authentication

Describes how to pass a credential objects to `UTL_SMTP.SET_CREDENTIAL` APIs.

The `SET_CREDENTIAL` subprogram sends the AUTH command to authenticate to the SMTP server.

The `UTL_SMTP.SET_CREDENTIAL` subprogram enables you to pass credential objects to set SMTP authentication. Credential objects are schema objects, hence they can be accessed only by privileged users and enable you to configure schema-level privileges to access control the credentials. Passing credential objects is a appropriate and secure way to store and manage username/password/keys for authentication.

The `UTL_SMTP.SET_CREDENTIAL` subprogram is a secure and convenient alternative to `UTL_SMTP.AUTH` subprogram.

Example

```
...
UTL_SMTP.AUTH (l_mail_conn, 'ocid1.user.oc1.username', 'xxxxxxxxxxxx',
schemes => 'PLAIN');
...
```

As shown in the example above, when you invoke `AUTH` subprogram, you must pass the username/password in clear text as part of PL/SQL formal parameters. You might need to embed the username/password into various PL/SQL automation or cron scripts. Passing clear text passwords is a compliance issue that is addressed in `UTL_SMTP.SET_CREDENTIAL` subprogram.

See AUTH Function and Procedure for more information.

`UTL_SMTP.SET_CREDENTIAL` Syntax

```
PROCEDURE UTL_SMTP.SET_CREDENTIAL (
    c          IN OUT NOCOPY connection,
    credential IN            VARCHAR2,
    schemes    IN            VARCHAR2 DEFAULT NON_CLEARTEXT_PASSWORD_SCHEMES
);

FUNCTION UTL_SMTP.SET_CREDENTIAL (
    c          IN OUT NOCOPY connection,
    credential IN            VARCHAR2,
```

```
    schemes    IN            VARCHAR2 DEFAULT NON_CLEARTEXT_PASSWORD_SCHEMES)
    RETURN reply;
```

Example

- Create a credential object:

```
BEGIN DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name => 'HTTP_CRED',
    username        => 'web_app_user',
    password        => '<password>' );
END;
```

This creates a credential object which creates a stored username/password pair.

See CREATE_CREDENTIAL Procedure for more information.

See Specifying Scheduler Job Credentials for more information.

- Execute `UTL_SMTP.SET_CREDENTIAL` procedure:

```
DECLARE
      l_mail_conn UTL_SMTP.CONNECTION;
    BEGIN
      l_mail_conn := UTL_SMTP.OPEN_CONNECTION('smtp.example.com', 587);
      UTL_SMTP.SET_CREDENTIAL(l_mail_conn, 'SMTP_CRED', SCHEMES =>
'PLAIN');
      ...
END;
```

This example sends the command to authenticate to the SMTP server. The Web server needs this information to authorize the request. The value `l_mail_conn` is the SMTP connection, `SMTP_CRED` is the credentials name and `PLAIN` is the SMTP authentication scheme.

See UTL_SMTP for more information.

See PL/SQL Packages Notes for Autonomous Database for information on restrictions for `UTL_SMTP` on Autonomous Database.

Send Email from Autonomous Database Using DBMS_CLOUD_NOTIFICATION

Use the `DBMS_CLOUD_NOTIFICATION` package to send messages and query results as email.

> **Note:**
>
> `DBMS_CLOUD_NOTIFICATION` supports sending email only to public SMTP endpoints. Oracle Cloud Infrastructure Email Delivery Service is the only supported email provider.

- Send Messages as Email from Autonomous Database
  You can use the `DBMS_CLOUD_NOTIFICATION` to send messages as an email.

- Send Query Results as Email from Autonomous Database
  You can use the `DBMS_CLOUD_NOTIFICATION` package to send the results of a query as an email.

Send Messages as Email from Autonomous Database

You can use the `DBMS_CLOUD_NOTIFICATION` to send messages as an email.

> **Note:**
>
> You can only use `DBMS_CLOUD_NOTIFICATION` for Email notifications with Autonomous Database version 19.21 and above.

1. Identify your SMTP connection endpoint for Email Delivery. You may need to subscribe to additional Oracle Cloud Infrastructure regions if Email Delivery is not available in your current region.

   For example, select one of the following for the SMTP connection endpoint:

   - smtp.us-phoenix-1.oraclecloud.com
   - smtp.us-ashburn-1.oraclecloud.com
   - smtp.email.uk-london-1.oci.oraclecloud.com
   - smtp.email.eu-frankfurt-1.oci.oraclecloud.com

   See Configure SMTP Connection for more information.

2. Generate SMTP credentials for Email Delivery. `UTL_SMTP` uses credentials to authenticate with Email Delivery servers when you send email.

   See Generate SMTP Credentials for a User for more information.

3. Create an approved sender for Email Delivery. Complete this step for all email addresses you use as the "`From`" with `UTL_SMTP.MAIL`.

   See Managing Approved Senders for more information.

4. Create a credential object and use `DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE` to send a message as an email.

```
BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'EMAIL_CRED',
        username        => 'username',
        password        => 'password'
   );
   END;
/
BEGIN
   DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
        provider        => 'email',
        credential_name => 'EMAIL_CRED',
        message         => 'Subject content',
        params          => json_object('recipient' value
'mark@example.com, suresh@example.com',
                                       'to_cc'  value 'nicole@example.com,
jordan@example.com',
```

```
                                                'to_bcc' value
'manisha@example.com',
                                                'subject' value 'Test subject',
                                                'smtp_host' value
'smtp.email.example.com',
                                                'sender'    value
'approver_sender@example.com' )
   );
   END;
/
```

Use the `params` parameter to specify the sender, smtp_host, subject, recipient, and recipients of a CC or BCC in `string` values.

- *sender*: specifies the Email ID of the approved sender from Step 3.
- *smtp_host*: specifies the SMTP host name from step 2.
- *subject*: specifies the subject of the email.
- *recipient*: specifies the email IDs of recipients. Use a comma between email IDs when there are multiple recipients.
- *to_cc*: specifies the email IDs that are receiving a CC of the email. Use a comma between email IDs when there are multiple CC recipients.
- *to_bcc*: specifies the email IDs that are receiving a BCC of the email. Use a comma between email IDs when there are multiple BCC recipients.

See SEND_MESSAGE Procedure for more information.

Send Query Results as Email from Autonomous Database

You can use the `DBMS_CLOUD_NOTIFICATION` package to send the results of a query as an email.

To use `DBMS_CLOUD_NOTIFICATION` to send mail:

1. Identify your SMTP connection endpoint for Email Delivery. You may need to subscribe to additional Oracle Cloud Infrastructure regions if Email Delivery is not available in your current region.

   For example, select one of the following for the SMTP connection endpoint:

   - smtp.us-phoenix-1.oraclecloud.com
   - smtp.us-ashburn-1.oraclecloud.com
   - smtp.email.uk-london-1.oci.oraclecloud.com
   - smtp.email.eu-frankfurt-1.oci.oraclecloud.com

   See Configure SMTP Connection for more information.

2. Generate SMTP credentials for Email Delivery. `UTL_SMTP` uses credentials to authenticate with Email Delivery servers when you send email.

   See Generate SMTP Credentials for a User for more information.

3. Create an approved sender for Email Delivery. Complete this step for all email addresses you use as the "`From`" with `UTL_SMTP.MAIL`.

   See Managing Approved Senders for more information.

4. Create a credential object and use `DBMS_CLOUD_NOTIFICATION.SEND_DATA` to send the output of a query as an email.

```
BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'EMAIL_CRED',
        username        => 'username',
        password        => 'password'
   );
   END;
/
BEGIN
   DBMS_CLOUD_NOTIFICATION.SEND_DATA(
        provider        => 'email',
        credential_name => 'EMAIL_CRED',
        query           => 'SELECT tablespace_name FROM dba_tablespaces',
        params          => json_object('recipient' value
'mark@example.com, suresh@example.com',
                                       'to_cc'  value
'nicole@example.com1, jordan@example.com',
                                       'to_bcc' value
'manisha@example.com',
                                       'subject' value 'Test subject',
                                       'type' value 'json',
                                       'title' value 'mytitle',
                                       'message' value 'This is the
message',
                                       'smtp_host' value
'smtp.email.example.com',
                                       'sender'    value
'approver_sender@example.com' )
   );
   END;
/
```

Use the `params` parameter to specify the sender, smtp_host, subject, recipient, recipients of a CC or BCC, the message, data type, and the title in `String` values.

- *sender*: specifies the Email ID of the approved sender from Step 3.
- *smtp_host*: specifies the SMTP host name from step 2.
- *subject*: specifies the subject of the email. The maximum size is 100 characters.
- *recipient*: This specifies the email IDs of recipients. Use a comma between email IDs when there are multiple recipients.
- *to_cc*: specifies the email IDs that are receiving a CC of the email. Use a comma between email IDs when there are multiple CC recipients.
- *to_bcc*: specifies the email IDs that are receiving a BCC of the email. Use a comma between email IDs when there are multiple BCC recipients.
- *message*: specifies the message text.
- *type*: specifies the output format as either CSV or JSON.
- *title*: specifies the title of the attachment of SQL output. The title should only contain letters, digits, underscores, hyphens, or dots as characters in its value due to it being used to generate a file name.

Notes for sending the results of a query as mail:

- You can only use `DBMS_CLOUD_NOTIFICATION` for mail notifications with Autonomous Database version 19.21 and above.
- The maximum message size for use with `DBMS_CLOUD_NOTIFICATION.SEND_DATA` for mail notification is 32k bytes.

See SEND_DATA Procedure for more information.

Send Slack Notifications from Autonomous Database

Describes how to configure Slack so that you can send messages, alerts, or output of a query from Autonomous Database to a Slack Channel. Also describes the procedures you use to send Slack notifications.

- Prepare to Send Slack Notifications from Autonomous Database
  To send Slack notifications, you must configure your Slack application to receive messages from Autonomous Database. Next, create a credential to use with the `DBMS_CLOUD_NOTIFICATION` procedures to send Slack notifications from Autonomous Database.
- Send Messages to a Slack Channel
- Send Query Results to a Slack Channel

Prepare to Send Slack Notifications from Autonomous Database

To send Slack notifications, you must configure your Slack application to receive messages from Autonomous Database. Next, create a credential to use with the `DBMS_CLOUD_NOTIFICATION` procedures to send Slack notifications from Autonomous Database.

To use Slack with `DBMS_CLOUD_NOTIFICATION` procedures:

1. Create your Slack app.

   The Slack app is installed in a Slack Workspace, which in turn has Channels where messages can be sent. The bot token of the Slack app must have the following permission scopes defined:

   ```
   channels:read
   chat:write
   files:write
   ```

   See Creating an app for information on setting up a Slack app.
2. Have your Slack admin add the Slack app to the channels to which `DBMS_CLOUD_NOTIFICATION` can send message through the "Integrations" option in the channel.

   See Basic app setup for more information.
3. Locate the app's bot token available in the Slack app specific page, under `https://app.slack.com`.

   See Basic app setup for more information.
4. Create a credential object to access the Slack app from Autonomous Database.

> **Tip:**
>
> If you can not use the `CREATE_CREDENTIAL` procedure successfully, consult with the ADMIN user to grant execute access on `DBMS_CLOUD` packages.

The credential's username is `SLACK_TOKEN` and the password is the bot token.

For example:

```
BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
     credential_name => 'SLACK_CRED',
     username    => 'SLACK_TOKEN',
     password    => 'xoxb-34....96-34....52-zW....cy');
END;
/
```

See CREATE_CREDENTIAL Procedure for more information.

5. Configure access control to allow user access to external network services (Slack).

```
BEGIN
 DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE (
  host         => 'slack.com',
  lower_port   => 443,
  upper_port   => 443,
  ace          => xs$ace_type(
      privilege_list => xs$name_list('http'),
      principal_name => example_invoking_user,
      principal_type => xs_acl.ptype_db));
END;
```

See Configuring Access Control for External Network Services for more information.

Send Messages to a Slack Channel

After creating the Slack credential object as described in Prepare to Send Slack Notifications from Autonomous Database, you can use the `DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE` procedure to send a message to a Slack channel.

Example:

```
BEGIN
   DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
      provider          => 'slack',
      credential_name   => 'SLACK_CRED',
      message           => 'Alert from Autonomous Database...',
      params            => json_object('channel' value 'C0....08'));
END;
/
```

Use the `params` parameter to specify the Slack channel.

- `channel`: Specifies the Channel ID.

The Channel ID is a unique ID for a channel and is different from the channel name. In Slack, when you view channel details you can find the Channel ID on the “About” tab. See How to Find your Slack Team ID and Slack Channel ID for more information.

See SEND_MESSAGE Procedure for more information.

Send Query Results to a Slack Channel

After creating the Slack credential object as described in Prepare to Send Slack Notifications from Autonomous Database, you can use the `DBMS_CLOUD_NOTIFICATION.SEND_DATA` procedure to send the output of a query to a Slack channel.

Example:

```
BEGIN
  DBMS_CLOUD_NOTIFICATION.SEND_DATA(
      provider => 'slack',
      credential_name => 'SLACK_CRED',
      query => 'SELECT username, account_status, expiry_date FROM USER_USERS
WHERE rownum < 5',
      params => json_object('channel' value 'C0....08',
                            'type' value 'csv'));
END;
/
```

Use the `params` parameter to specify the Slack channel and the data type:

- `channel`: Specifies the Channel ID.

  The Channel ID is a unique ID for a channel and is different from the channel name. In Slack, when you view channel details you can find the Channel ID on the “About” tab. See How to Find your Slack Team ID and Slack Channel ID for more information.

- `type`: Specifies the output type. Valid values are: '`csv`' or '`json`'.

See SEND_DATA Procedure for more information.

Related Topics

- SEND_DATA Procedure
  The `SEND_DATA` procedure sends the results of the specified query to a provider.

Send Microsoft Teams Notifications from Autonomous Database

Describes how to configure Microsoft Teams so that you can send messages, alerts, or output of a query from Autonomous Database to a Microsoft Teams Channel. Also describes the procedures you use to send Microsoft Teams notifications.

- Prepare to Send Microsoft Teams Notifications from Autonomous Database
  Get started by configuring a bot in your Microsoft Teams app. Next, create a credential to use with the `DBMS_CLOUD_NOTIFICATION` procedures to send Microsoft Teams notifications from Autonomous Database.
- Send Messages to a Microsoft Teams Channel
- Send Query Results to a Microsoft Teams Channel

Prepare to Send Microsoft Teams Notifications from Autonomous Database

Get started by configuring a bot in your Microsoft Teams app. Next, create a credential to use with the `DBMS_CLOUD_NOTIFICATION` procedures to send Microsoft Teams notifications from Autonomous Database.

To configure Microsoft Teams notifications:

1. Create your Microsoft Teams app and add a bot in it. See Developer Portal for Teams for information on setting up an app.
2. From the **Bot Management** section, ensure that the bot has a secret key, scope set to *Team* and permission to send notifications.
3. Publish the app to your org to make it available to the people in your org.
4. After your IT admin approves the app from the admin center, install the app from the **Apps** section in Teams.
5. Request the `Files.ReadWrite.All` and `ChannelSettings.Read.All` permissions to the app for Graph API from the Azure Portal using the following instructions:
   a. Log into your Azure Portal, navigate to **Azure Active Directory** using the left panel, and select the **Application** option.
   b. The Application page displays the apps that you own along with the bots added to those apps. Click the bot to view its details.
   c. Copy the *directory/tenant id* from the bot overview page for later use.
   d. Then, go to **API permissions** in the left panel. Under **API permissions**, click **Add permission**, select **Microsoft graph** and then **Application permission**.
   e. Search for `Files.ReadWrite.All` and `ChannelSettings.Read.All` permissions and add them.
6. Have your IT admin approve the above-requested permissions from the Azure portal by following the steps below:
   a. Log into your Azure Portal, navigate to **Azure Active Directory** using the left panel, and select the **Application** option.
   b. Select **All applications** from the **Application** page.
   c. Search the *application/bot* by its name, go to **API permissions**, and grant the ADMIN consent for the requested permissions: `Files.ReadWrite.All` and `ChannelSettings.Read.All`.

   > **Tip:**
   >
   > After your app is approved by the IT Admin, you can provide the bot id and secret key to other users to install the app within Teams in the org.

7. After the app is approved by the IT admin and the above requested permissions are granted, you can use the app's bot ID and secret key are used to create the credential object and generate a bot token.
8. To send a query result to a Microsoft Teams channel, obtain the `team id` and the `tenant id`.

> **Tip:**
>
> The `team id` is located in the team link between /team/ and /conversations. The `tenant id` is located after "tenantId=" at the end of the team link. This link is found by clicking the three dots next to the team name and selecting **Get link to team**.

For example:

```
https://teams.microsoft.com/l/team/teamID/conversations?
groupId=groupid%tenantId=tenantid
```

9. Obtain the `channelID`.

> **Tip:**
>
> `channelID` is located in the channel link between /team/ and the channel name. This link is found by clicking the three dots next to the channel name and selecting **Get link to channel**.

For example:

```
https://teams.microsoft.com/l/channel/channelID/channel_name?
groupId=groupid&tenantId=tenantid
```

10. Create a credential object to access the Microsoft Teams app from Autonomous Database.

> **Tip:**
>
> If you can not use the `CREATE_CREDENTIAL` procedure successfully, consult with the ADMIN user to grant execute access on `DBMS_CLOUD` packages.

The credential's username is the bot_id and the password is the bot key.

For example:

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(credential_name => 'TEAMS_CRED',
        username        => 'bot_id',
        password        => 'bot_secret');
END;
/
```

See CREATE_CREDENTIAL Procedure for more information.

Send Messages to a Microsoft Teams Channel

After creating the Microsoft Teams credential object as described in Prepare to Send Microsoft Teams Notifications from Autonomous Database, you can use the

`DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE` procedure to send a message to a Microsoft Teams channel.
Example:

```
BEGIN
     DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
        provider        => 'msteams',
        credential_name => 'TEAMS_CRED',
        message         => 'text from new teams api',
        params          => json_object('channel' value 'channelID'));
END;
/
```

Use the `params` parameter to specify the channel.

- *channel*: specifies the Channel ID obtained from Step 10 in Prepare to Send Microsoft Teams Notifications from Autonomous Database.

See SEND_MESSAGE Procedure for more information.

Send Query Results to a Microsoft Teams Channel

After creating the Microsoft Teams credential object as described in Prepare to Send Microsoft Teams Notifications from Autonomous Database, you can use the `DBMS_CLOUD_NOTIFICATION.SEND_DATA` procedure to send the output of a query to a Microsoft Teams channel.
Example:

```
BEGIN
     DBMS_CLOUD_NOTIFICATION.SEND_DATA(provider => 'msteams',
        credential_name => 'TEAMS_CRED',
        query           => 'SELECT tablespace_name FROM dba_tablespaces',
        params          => json_object('tenant'value '5b743bc******c0286',
                                       'team'value '0ae401*********5d2bd',
                                       'channel'value
'19%3a94be023*****%40thread.tacv2',
                                       'title'value 'today',
                                       'type'value 'csv'));
END;
/
```

Use the `params` parameter to specify the tenant, team, channel, title, and data type in `string` values.

- *tenant*: specifies the tenant ID obtained from Step 8 in Prepare to Send Microsoft Teams Notifications from Autonomous Database.
- *team*: specifies the team ID obtained from Step 8 in Prepare to Send Microsoft Teams Notifications from Autonomous Database.
- *channel*: specifies the channel ID obtained from Step 9 in Prepare to Send Microsoft Teams Notifications from Autonomous Database.
- *title*: specifies the title of the file. The title can only contain alphabets, digits, underscores, and hyphens. The file name that appears in Microsoft Teams will be a concatenation of the

title parameter and the timestamp to ensure uniqueness. The maximum title size is 50 characters.

For example: '*title*'_'*timestamp*'.'*format*'

- *type*: This specifies the output format. Valid values are CSV or JSON.

> **Note:**
>
> The maximum file size supported when using `DBMS_CLOUD_NOTIFICATION.SEND_DATA` for Microsoft Teams is 4MB.

See SEND_DATA Procedure for more information.

User Defined Notification Handler for Scheduler Jobs

Database Scheduler provides an email notification mechanism to track the status of periodically running or automated jobs. In addition to this, the Database Scheduler also supports user-defined PL/SQL Scheduler job notification handler procedure.

Adding a scheduler job notification handler procedure allows you to monitor scheduled or automated jobs running in your Autonomous Database.

- About User Defined Notification Handler for Scheduler Jobs
  The Database Scheduler supports job notification handler procedure that can make use of custom code to call HTTP or REST endpoints for improved monitoring of scheduler jobs in an Autonomous Database instance.
- Create a Job Notification Handler Procedure
  Provides steps to create a job notification handler.
- Register the Job Handler Notification Procedure
  Use `DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE` procedure to set `JOB_NOTIFICATION_HANDLER` attribute value to register the job handler notification procedure.
- Trigger the Job Handler Notification Procedure
  You must call the `DBMS_SCHEDULER.ADD_JOB_EMAIL_NOTIFICATION` procedure to trigger the user defined job notification handler procedure.
- De-Register the Job Handler Notification Procedure
  Use `DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE` to de-register the job handler notification procedure.

About User Defined Notification Handler for Scheduler Jobs

The Database Scheduler supports job notification handler procedure that can make use of custom code to call HTTP or REST endpoints for improved monitoring of scheduler jobs in an Autonomous Database instance.

The handler procedure receives all pertinent information regarding the job, such as the job owner's name, class name, event type, and timestamp in JSON format. Based on the information, the handler procedure then takes the required action.

See DBMS_SCHEDULER for more information on Oracle Scheduler.

Configuring user defined notification handler for scheduler jobs consists of these steps:

- Create a job notification handler procedure as described in: Create a Job Notification Handler Procedure.
- Register the job notification handler procedure for the database as described in: Register the Job Handler Notification Procedure.
- Trigger the job notification handler procedure as described in: Trigger the Job Handler Notification Procedure
- De-Register the job notification handler procedure for the database as described in: De-Register the Job Handler Notification Procedure.

Create a Job Notification Handler Procedure

Provides steps to create a job notification handler.

1. Create a credential object.

   See CREATE_CREDENTIAL Procedure for more information.

   See Specifying Scheduler Job Credentials for more information.

2. Create a Job notification handler:

   Example to create a job notification handler procedure to send a message to a Slack channel:

```
CREATE OR REPLACE PROCEDURE ADMIN.SEND_NOTIFICATION(data_in CLOB) AS
BEGIN
  DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE (
    provider          => 'slack',
    credential_name   => 'SLACK_CRED',
    message           => data_in,
    params            => JSON_OBJECT('channel' value 'adw-job-nfy');
end;
/
```

   This example creates the `SEND_NOTIFICATION` procedure.

   This user defined procedure sends the provided input data as a message to the specified Slack channel.

   See Send Slack Notifications from Autonomous Database for more information.

   See SEND_MESSAGE Procedure for more information.

   Example to create a job notification handler procedure to insert the message into a table:

```
CREATE TABLE ADMIN.JOB_STATUS(jnfy_data CLOB);

CREATE OR REPLACE PROCEDURE ADMIN.INSERT_JOB_STATUS(data_in CLOB) AS
  l_sessuser VARCHAR2(128) := SYS_CONTEXT('userenv','session_user');
BEGIN
  INSERT INTO ADMIN.JOB_STATUS (jnfy_data) VALUES (data_in || TO_CLOB(' :
Sent By Session User : ' || l_sessuser));
  COMMIT;
END;
/
```

This example creates the JOB_STATUS table and INSERT_JOB_STATUS procedure to insert the session-specific values into the table.

You must be logged in as the ADMIN user or have CREATE ANY PROCEDURE system privilege to create a job notification handler procedure.

> **Note:**
>
> An ORA-27405 is returned when you specify an invalid owner or object name as the job notification handler procedure.

Existing DBMS_SCHEDULER procedures ADD_JOB_EMAIL_NOTIFICATION and REMOVE_JOB_EMAIL_NOTIFICATION are enhanced to support the job notification handler procedure.

See ADD_JOB_EMAIL_NOTIFICATION Procedure and REMOVE_JOB_EMAIL_NOTIFICATION Procedure for more information.

Use DBA_SCHEDULER_NOTIFICATIONS dictionary view to query the list of notifications for a scheduler job. See DBA_SCHEDULER_NOTIFICATIONS for more information.

Register the Job Handler Notification Procedure

Use DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE procedure to set JOB_NOTIFICATION_HANDLER attribute value to register the job handler notification procedure.

The JOB_NOTIFICATION_HANDLER attribute specifies the job handler notification procedure that you want to use.

To register the job handler notification procedure you must:

- Be logged in as the ADMIN user or have MANAGE SCHEDULER privilege.
- Have EXECUTE privilege on the handler procedure or EXECUTE ANY PROCEDURE system privilege.

The JOB_NOTIFICATION_HANDLER attribute and EMAIL_SERVER attribute are mutually exclusive. The ATTRIBUTE parameter of the SET_SCHEDULER_ATTRIBUTE procedure can have either the JOB_NOTIFICATION_HANDLER or the EMAIL_SERVER value at a time. You are allowed to either configure email notifications or create your notification handler for your scheduler jobs.

An ORA-27488 error is raised when you attempt to set both EMAIL_SERVER and JOB_NOTIFICATION_HANDLER global attributes.

Execute DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE procedure to register the job handler notification procedure:

```
BEGIN

DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE('job_notification_handler','ADMIN.SEND_
NOTIFICATION');
END;
/
```

This example registers the ADMIN.SEND_NOTIFICATION procedure as the job handler notification procedure for your database.

See SET_SCHEDULER_ATTRIBUTE Procedure for more information.

Execute this command to verify the job notification handler:

```
SELECT value FROM dba_scheduler_global_attribute WHERE
attribute_name='JOB_NOTIFICATION_HANDLER';

VALUE
---------------
"ADMIN"."SEND_NOTIFICATION"
```

See DBA_SCHEDULER_GLOBAL_ATTRIBUTE for more information.

You must assign `EXECUTE` privilege to allow other users to use the job notification handler. For example:

```
GRANT EXECUTE ON ADMIN.SEND_NOTIFICATION To DWUSER;
```

`ORA-27476 ("\"%s\".\"%s\" does not exist")` or `ORA-27486 ("insufficient privileges")` error is thrown if you do not have privilege on the job handler notification procedure.

Trigger the Job Handler Notification Procedure

You must call the `DBMS_SCHEDULER.ADD_JOB_EMAIL_NOTIFICATION` procedure to trigger the user defined job notification handler procedure.

The overloaded form of the `DBMS_SCHEDULER.ADD_JOB_EMAIL_NOTIFICATION` enables you to trigger the job notification handler procedure and send a notification. However, these notifications are not sent out in the form of an email when you have registered the job notification handler procedure. Hence, the parameters `SUBJECT`, and `BODY` are optional. The `RECIPIENT` parameter is still mandatory. Since this overloaded form of procedure is not sending email notifications so, you can provide any string value for the `RECIPIENT` parameter.

For example, the following steps create a scheduler job, add notification for the job, enable the job, verify the notification entries, show the data received by the job notification handler procedure, and remove notification for the job.

1. Create a scheduler job in the `DWUSER` schema:

```
BEGIN
 DBMS_SCHEDULER.CREATE_JOB(
   job_name   => 'DWUSER.MY_JOB',
   job_type   => 'PLSQL_BLOCK',
   job_action => 'BEGIN null; END;',
   enabled    => FALSE,
   auto_drop  => FALSE);
END;
/
```

This creates a job `DWUSER.MY_JOB` with the specified attributes.

See CREATE_JOB Procedure for more information.

2. Configure the job to send notifications at specified events.

```
BEGIN
  DBMS_SCHEDULER.ADD_JOB_EMAIL_NOTIFICATION(
    job_name   => 'DWUSER.MY_JOB',
    recipients => 'PLACEHOLDER_STRING',
    subject    => 'Job Notification-%job_owner%.%job_name%-%event_type%',
    body       => '%event_type% occurred at %event_timestamp%.
%error_message%',
    events     => 'job_started, job_succeeded, job_completed');
END;
/
```

This procedure adds notifications for the `DWUSER.MY_JOB` job. The notifications are sent whenever any of the specified job state events is raised.

See ADD_JOB_EMAIL_NOTIFICATION Procedure for more information.

3. Enable the scheduler job.

```
EXEC DBMS_SCHEDULER.ENABLE('DWUSER.MY_JOB');
```

See ENABLE Procedure for more information.

When you enable the `DWUSER.MY_JOB` job, the `USER_SCHEDULER_NOTIFICATIONS` view is populated with the job notification entries. To verify you can query the `USER_SCHEDULER_NOTIFICATIONS` view. For example:

```
SELECT  job_name, recipient, event, subject, body
FROM user_scheduler_notifications
ORDER BY notification_owner, owner, job_name;

JOB_NAME    RECIPIENT            EVENT          SUBJECT         BODY
-------    ------------------   -------------  ---------------
---------------
MY_JOB     placeholder_string   JOB_STARTED    Job Notificatio
%event_type% oc
                                               n-%job_owner%.% curred at
%even
                                               job_name%-%even
t_timestamp%. %
                                               t_type%
error_message%

MY_JOB     placeholder_string  JOB_SUCCEEDED    Job Notificatio
%event_type% oc
                                                n-%job_owner%.% curred at
%even
                                                job_name%-%even
t_timestamp%. %
                                                t_type%
error_message%

MY_JOB    placeholder_string  JOB_COMPLETED    Job Notificatio
%event_type% oc
                                               n-%job_owner%.% curred at
```

```
%even
                                                    job_name%-%even
t_timestamp%. %
                                                    t_type%
error_message%
```

See USER_SCHEDULER_NOTIFICATIONS View for more information.

When the job `DWUSER.MY_JOB` is executed and any of the specified job state events is raised, the job notification handler procedure is triggered and receives the specified information as input. For example, the `ADMIN.SEND_NOTIFICATION` job notification handler procedure receives the following:

```
{"job_owner":"DWUSER","job_name":"MY_JOB","job_class_name":"DEFAULT_JOB_CLA
SS","event_type":"JOB_STARTED","event_timestamp":"12-JAN-23 08.13.46.193306
 PM UTC","error_code":0,"error_msg":null,"sender":null,"recipient":"data_lo
ad_pipeline","subject":"Job Notification-DWUSER.MY_JOB-JOB_STARTED","msg_te
xt":"JOB_STARTED occurred at 12-JAN-23 08.13.46.193306 PM UTC. ","comments"
:"User defined job notification handler"}

{"job_owner":"DWUSER","job_name":"MY_JOB","job_class_name":"DEFAULT_JOB_CLA
SS","event_type":"JOB_SUCCEEDED","event_timestamp":"12-JAN-23 08.13.46.2863
44 PM UTC","error_code":0,"error_msg":null,"sender":null,"recipient":"data_
load_pipeline","subject":"Job Notification-DWUSER.MY_JOB-JOB_SUCCEEDED","ms
g_text":"JOB_SUCCEEDED occurred at 12-JAN-23 08.13.46.286344 PM UTC. ","com
ments":"User defined job notification handler"}
```

4. Remove the job notification. For example:

```
EXEC DBMS_SCHEDULER.REMOVE_JOB_EMAIL_NOTIFICATION('DWUSER.MY_JOB');
```

See REMOVE_JOB_EMAIL_NOTIFICATION Procedure for more information.

De-Register the Job Handler Notification Procedure

Use `DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE` to de-register the job handler notification procedure.

To de-register the job handler notification procedure you must be logged in as the ADMIN user or have `MANAGE SCHEDULER` privilege.

Example to de-register the job handler notification procedure:

```
BEGIN
  DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE ('job_notification_handler','');
END;
/
```

Oracle Extensions for IDEs

Oracle extensions let developers connect to, browse, and manage Autonomous Databases directly from common IDEs.

- Use Oracle Cloud Infrastructure Toolkit for Eclipse
  Oracle Cloud Infrastructure Toolkit for Eclipse is a plugin that enables Java developers to easily connect to Oracle Autonomous Database through their IDE. The plugin is free and is available for Linux, UNIX, Microsoft Windows, and Apple Mac OS.
- Use Oracle Developer Tools for Visual Studio
  Oracle Developer Tools for Visual Studio is a tightly integrated extension for Microsoft Visual Studio and Oracle Autonomous Database. The extension is free and supports Visual Studio 2019 and Visual Studio 2017 on Microsoft Windows.
- Use Oracle Developer Tools for VS Code
  Oracle Developer Tools for VS Code is a tightly integrated extension for Microsoft Visual Studio Code (VS Code) and Oracle Autonomous Database. The extension is free and is available for Linux, Microsoft Windows, and Apple Mac OS.

Use Oracle Cloud Infrastructure Toolkit for Eclipse

Oracle Cloud Infrastructure Toolkit for Eclipse is a plugin that enables Java developers to easily connect to Oracle Autonomous Database through their IDE. The plugin is free and is available for Linux, UNIX, Microsoft Windows, and Apple Mac OS.

You can use the plugin to perform cloud and database operations right from Eclipse, such as creating Autonomous Databases, stopping and starting, scaling up and down, and so on. You can also use the plugin to easily connect to the databases to browse the schema, access tables, execute SQL statements, and perform other development tasks.

Users with permissions to manage the databases can perform a number of actions, including those listed below. For detailed information about permissions, see Toolkit for Eclipse in the Oracle Cloud Infrastructure documentation. You can:

- Create Autonomous Databases
- Start, stop, terminate, clone, and restore Autonomous Databases
- Scale up and down
- Download the client credentials zip file (database wallet)
- Connect to Autonomous Databases
- Browse the schema
- Choose compartments and regions
- Change the administrator password, update the license type, and so on

Download the latest version of the plugin from GitHub (`com.oracle.oci.eclipse-`*`version`*`.zip`, where *`version`* is the latest version, for instance `1.2.0`):

`https://github.com/oracle/oci-toolkit-eclipse/releases`

Then follow the installation instructions and details about how to get started in this step-by-step walkthrough:

New Eclipse Plugin for Accessing Autonomous Database (ATP/ADW)

Use Oracle Developer Tools for Visual Studio

Oracle Developer Tools for Visual Studio is a tightly integrated extension for Microsoft Visual Studio and Oracle Autonomous Database. The extension is free and supports Visual Studio 2019 and Visual Studio 2017 on Microsoft Windows.

You can use the extension to perform database management operations right from Visual Studio, such as creating Autonomous Databases, stopping and starting, scaling up and down, and so on. You can also use the extension to easily connect to the databases and perform development tasks, such as browsing your Oracle schema and launching integrated Oracle designers and wizards to create and alter schema objects.

Users with permissions to manage the databases can perform a number of actions, including the following:

- Sign up for Oracle Cloud
- Connect to a cloud account using a simple auto-generated config file and key file
- Create new or clone existing Always Free Autonomous Database, Autonomous Database Dedicated, and Autonomous Database Serverless databases
- Automatically download credentials files (including wallets) and quickly connect, browse, and operate on Autonomous Database schemas
- Change compartments and regions without reconnecting
- Start, stop, or terminate Autonomous Database
- Scale up/down Autonomous Database resources
- Restore from backup
- Update instance credentials, update the license type used
- Rotate wallets
- Convert Always Free Autonomous Database into paid databases

> **Note:**
>
> Promotion of Always Free to paid is supported only if the Always Free instance has database release *19c*.

Download the extension from Visual Studio Marketplace:

- Oracle Developer Tools for Visual Studio 2019
- Oracle Developer Tools for Visual Studio 2017

You'll find lots of information about the extension on those Marketplace pages.

Then follow the installation instructions and details about how to get started in this step-by-step walkthrough:

New Release: Visual Studio Integration with Oracle Autonomous Database

For detailed information about how to use the extension, see the online documentation that's optionally installed with Oracle Developer Tools for Visual Studio. Press the F1 key to display the context-sensitive help for each dialog.

Use Oracle Developer Tools for VS Code

Oracle Developer Tools for VS Code is a tightly integrated extension for Microsoft Visual Studio Code (VS Code) and Oracle Autonomous Database. The extension is free and is available for Linux, Microsoft Windows, and Apple Mac OS.

You can use the extension to connect to Autonomous Databases right from Visual Studio Code and easily explore database schema, view table data, and edit and execute SQL and PL/SQL.

Download the extension from Visual Studio Marketplace:

Oracle Developer Tools for VS Code

Installation instructions and information about how to get started can be found in this quick start guide:

Getting Started Using Oracle Developer Tools for VS Code

Using Oracle Java on Autonomous Database

Autonomous Database supports Oracle JVM. Oracle JVM is a standard, Java-compatible environment that runs any pure Java application.

Oracle JVM is compatible with the standard JLS and the JVM specifications. It supports the standard Java binary format and the standard Java APIs. In addition, Oracle Database adheres to standard Java language semantics, including dynamic class loading at run time.

See About Using Java in Oracle Database for information on Oracle Java.

- Enable Oracle Java
  Use `DBMS_CLOUD_ADMIN.ENABLE_FEATURE` to enable Oracle Java on Autonomous Database.
- Check Oracle Java Version
  You can check the Oracle Java version and the component registry for information on Oracle Java in the Autonomous Database instance.
- Load Java classes and JAR Files into Autonomous Database
  You can use the client-side `loadjava` option to load Java classes and JAR files into Oracle JVM on an Autonomous Database instance.
- Notes for Oracle Java on Autonomous Database
  Provides notes for using Oracle Java on Autonomous Database.

Enable Oracle Java

Use `DBMS_CLOUD_ADMIN.ENABLE_FEATURE` to enable Oracle Java on Autonomous Database.

1. Run `DBMS_CLOUD_ADMIN.ENABLE_FEATURE`.

   ```
   BEGIN
      DBMS_CLOUD_ADMIN.ENABLE_FEATURE(
          feature_name => 'JAVAVM' );
   END;
   /
   ```

   This initiates the request to install Oracle Java on the Autonomous Database instance.

   See ENABLE_FEATURE Procedure for more information.

2. Restart the Autonomous Database instance.

   See Restart Autonomous Database for more information.

After you restart the Autonomous Database instance, Oracle JVM is enabled on the instance.

Check Oracle Java Version

You can check the Oracle Java version and the component registry for information on Oracle Java in the Autonomous Database instance.

1. Check the component registry status and version for Oracle Java.

```
SELECT status, version FROM DBA_REGISTRY
   WHERE comp_id = 'JAVAVM';

STATUS VERSION
------ ----------
VALID  19.0.0.0.0
```

   If Oracle Java is not installed, this query shows no rows.

2. Check the Oracle Java JDK version.

```
SELECT dbms_java.get_jdk_version FROM DUAL;

GET_JDK_VERSION
---------------
1.8.0_331
```

Load Java classes and JAR Files into Autonomous Database

You can use the client-side `loadjava` option to load Java classes and JAR files into Oracle JVM on an Autonomous Database instance.

The client-side `loadjava` option is supported as follows:

1. The JDK opens the JAR file.
2. The client-side opens a communication channel with the Autonomous Database.
3. The Java code is submitted to the Oracle JVM running on the Autonomous Database for loading.

This process is similar to creating a dynamic Java stored procedure from SQL code, where no file access is involved, but the code gets created.

Note the following:

- The Java code you load cannot invoke any operating system or network calls.
- Use of server-side `loadjava`, that is calls to the procedure `dbms_java.loadjava` is not supported. The procedure `dbms_java.loadjava` accesses the file system of the database server where the Oracle JVM runs. This is not allowed on Autonomous Database and calls to the procedure `dbms_java.loadjava` from within your Java application are not supported.

Notes for Oracle Java on Autonomous Database

Provides notes for using Oracle Java on Autonomous Database.

- You cannot disable Oracle Java after it is enabled on the Autonomous Database instance.
- Autonomous Database performs Oracle Java patching, as required, during the regular Autonomous Database maintenance window.

During Oracle Java patching, Java is not available and users could get an error similar to the following:

```
ERROR at line 1:
ORA-29548: Java system class reported: release of Java system classes in
the
database (19.0.0.0.220118 1.8) does not match that of the oracle executable
(19.0.0.0.220419 1.8).
```

During the maintenance window, when the Java patching phase is active there is no response for Java session calls or you see the `ORA-29548` error. After the maintenance window completes, Java usage is restored.

You can use the events `NewMaintenanceSchedule` and `ScheduledMaintenanceWarning` to be notified of Oracle Java patching. See Information Events on Autonomous Database for more information.

See About Scheduled Maintenance and Patching for more information.

Test Workloads with Oracle Real Application Testing

Oracle Real Application Testing is an extremely cost-effective and easy-to-use proactive performance management solution that enables businesses to fully assess the outcome of a system change in test or production.

- About Oracle Real Application Testing
  You can use Oracle Real Application Testing to capture a workload on the production system and replay it on a test system with the exact timing, concurrency, and transaction characteristics of the original workload.
- Capture-Replay Workloads between Autonomous Databases
  You can Capture and Replay from an Autonomous Database instance into another Autonomous Database instance.
- Capture-Replay Workloads between non-Autonomous and Autonomous Databases
  You can Capture and Replay from a non-Autonomous Database instance into an Autonomous Database.
- Test Workloads Against an Upcoming Patch
  Using the workload auto replay feature you can automatically capture a workload from a production database that is on the regular patch level and replay the workload on a target refreshable clone that is on the early patch level. This feature allows you to test an upcoming patch by running an existing workload that is in production against a patch before the patch reaches production.

About Oracle Real Application Testing

You can use Oracle Real Application Testing to capture a workload on the production system and replay it on a test system with the exact timing, concurrency, and transaction characteristics of the original workload.

- Real Application Testing enables you to test the effects of a system change on a workload without affecting the production system.
- Real Application Testing captures the workload on a production system and simulates the same workload on a test system.
- This provides an accurate method to test the impact of a variety of system changes.

You can use Oracle Database Replay to capture workload from an Autonomous Database instance as well as an on-premises database or any other cloud service database and replay it on Autonomous Database. This enables you to compare workloads between Autonomous Database, on-premises database, or other cloud service database and Autonomous Database.

Real Application Testing enables you to perform the following:

- Capture-Replay Workloads between Autonomous Databases

  See Capture-Replay Workloads between Autonomous Databases for more information.
- Capture a Workload from a non-Autonomous Database and Replay it on an Autonomous Database.

  See Capture-Replay Workloads between non-Autonomous and Autonomous Databases for more information.
- Capture a workload from the production database and replay it on the target database after a patch is applied to the target database.

  See Test Workloads Against an Upcoming Patch for more information.

Capture-Replay Workloads between Autonomous Databases

You can Capture and Replay from an Autonomous Database instance into another Autonomous Database instance.

This enables you to compare workloads across different Autonomous Database instances. These Autonomous Database instances may vary at patch levels, database versions, or regions.

The Capture-Replay workflow between Autonomous Databases consists of these steps:

- Subscribe to the Information event category on Oracle Cloud Infrastructure as described in:

  Subscribe to Information Events to be Notified of Capture and Replay Details.

  See Information Events on Autonomous Database for more information.
- Capture the workload on the production system as described in:

  Capture a Workload on an Autonomous Database Instance.
- Finish the current workload capture, as described in Finish a Workload Capture on Autonomous Database Instance.

  You can also cancel the current workload capture, as described in Cancel a Workload Capture on an Autonomous Database Instance.
- Prepare for a workload replay, on the test system, as described in Prepare a Workload Replay.
- Replay the captured workload, on the test system, as described in Replay a Workload on an Autonomous Database Instance.
- Subscribe to Information Events to be Notified of Capture and Replay Details
  You must subscribe to the `com.oraclecloud.databaseservice.autonomous.database.information` Information events to be notified at the start and completion of the capture and replay. These events also provide the PAR URL to the Object Storage to download the capture file and replay report.
- Capture a Workload
  The first step in using Database Replay is to capture the production workload.

- Cancel a Workload Capture on an Autonomous Database Instance
  Run `DBMS_CLOUD_ADMIN.CANCEL_WORKLOAD_CAPTURE` to cancel the current workload capture on your Autonomous Database instance.
- Finish a Workload Capture on Autonomous Database Instance
  Run `DBMS_CLOUD_ADMIN.FINISH_WORKLOAD_CAPTURE` to complete the current workload capture on your Autonomous Database instance.
- Prepare a Workload Replay
  Provide steps to prepare the refreshable clone for a replay.
- Replay a Workload on an Autonomous Database Instance
  After you complete a workload capture, you can replay it on a test system. Oracle replays the actions recorded during workload capture, with the same timing, concurrency, and transaction dependencies of the production system.

Subscribe to Information Events to be Notified of Capture and Replay Details

You must subscribe to the `com.oraclecloud.databaseservice.autonomous.database.information` Information events to be notified at the start and completion of the capture and replay. These events also provide the PAR URL to the Object Storage to download the capture file and replay report.

These Information events provide notifications about begin and end times of capture and replay as well as contain a PAR URL to capture and replay reports.

Autonomous Database Information events include the following:

- `WorkloadCaptureBegin`: This event is triggered when a workload capture is initiated.
- `WorkloadCaptureEnd`: This event is triggered when a workload capture completes successfully and generates a pre-authenticated (PAR) URL to download the capture file.
- `WorkloadReplayBegin`: This event is triggered when a workload replay is initiated.
- `WorkloadReplayEnd`: This event is triggered when a workload replay completes successfully and generates a pre-authenticated (PAR) URL to download the replay reports.

See Information Events on Autonomous Database for more information.

See Overview of Events for complete information on Oracle Cloud Infrastructure events.

Capture a Workload

The first step in using Database Replay is to capture the production workload.

When you begin workload capture on the production system, all requests from external clients directed to Oracle Database are tracked and stored in binary files called capture files.

A workload capture results in the creation of two subdirectories, `cap` and `capfiles`, which contain the capture files.

The capture files provide all pertinent information about the client request, including transaction details, bind values, and SQL text.

These capture files are platform independent and can be transported to another system.

- Capture a Workload on an Autonomous Database Instance
  Run `DBMS_CLOUD_ADMIN.START_WORKLOAD_CAPTURE` to initiate workload capture on your Autonomous Database instance.

Capture a Workload on an Autonomous Database Instance

Run `DBMS_CLOUD_ADMIN.START_WORKLOAD_CAPTURE` to initiate workload capture on your Autonomous Database instance.

You can capture a workload in an Autonomous Database instance and replay it in another Autonomous Database instance. You can replay the captured workload on the full or refreshable clone. The capture and replay targets must be in a consistent logical state.

See Cloning and Moving an Autonomous Database for more information.

To initiate a workload capture on your Autonomous Database instance, you must be logged in as the `ADMIN` user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

Example to initiate a workload capture:

```
BEGIN
   DBMS_CLOUD_ADMIN.START_WORKLOAD_CAPTURE(
        capture_name => 'test',
        duration     => 60);
   END;
/
```

This starts the workload capture on your Autonomous Database instance.

The parameters are:

- `capture_name`: is the name of the workload capture.
- `duration`: is the duration (in minutes) for which you need to capture the workload. This parameter is optional.

See START_WORKLOAD_CAPTURE Procedure for more information.

> **Note:**
>
> You must subscribe to the Information event `com.oraclecloud.databaseservice.autonomous.database.information` to be notified at the start of `START_WORKLOAD_CAPTURE`. See Information Events on Autonomous Database for more information.

You can find information about workload captures and replays in the `DBA_CAPTURE_REPLAY_HISTORY` view. See DBA_CAPTURE_REPLAY_HISTORY View for more information.

Cancel a Workload Capture on an Autonomous Database Instance

Run `DBMS_CLOUD_ADMIN.CANCEL_WORKLOAD_CAPTURE` to cancel the current workload capture on your Autonomous Database instance.

To cancel a workload capture, you must be logged in as the `ADMIN` user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

Example:

```
BEGIN
    DBMS_CLOUD_ADMIN.CANCEL_WORKLOAD_CAPTURE;
END;
/
```

This cancels the current workload capture and enables refresh on the refreshable clone.

You can query the `DBA_CAPTURE_REPLAY_STATUS` view to check the cancel workload status.

See DBA_CAPTURE_REPLAY_STATUS View for more information.

See CANCEL_WORKLOAD_CAPTURE Procedure for more information.

Finish a Workload Capture on Autonomous Database Instance

Run `DBMS_CLOUD_ADMIN.FINISH_WORKLOAD_CAPTURE` to complete the current workload capture on your Autonomous Database instance.

To stop a workload capture on your Autonomous Database instance, you must be logged in as the `ADMIN` user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

Example to stop a workload capture on your Autonomous Database instance:

```
BEGIN
    DBMS_CLOUD_ADMIN.FINISH_WORKLOAD_CAPTURE;
END;
/
```

To run this procedure you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

You can query the `DBA_CAPTURE_REPLAY_STATUS` view to check the status of a completed workload capture. See DBA_CAPTURE_REPLAY_STATUS View for more information.

You can query the `ID`, `NAME`, `START_TIME`, and `END_TIME` columns of the `DBA_WORKLOAD_CAPTURES` view to retrieve the details of your workload capture. See DBA_WORKLOAD_CAPTURES for more information.

The workload capture file is uploaded to the Object Store as a zip file.

> **Note:**
>
> You must subscribe to the Information event `com.oraclecloud.databaseservice.autonomous.database.information` to be notified about the completion of `FINISH_WORKLOAD_CAPTURE` as well as the Object Storage link to download the capture file. This PAR URL is contained in the `captureDownloadURL` field of the event and is valid for 7 days from the date of generation. See Information Events on Autonomous Database for more information.

See FINISH_WORKLOAD_CAPTURE Procedure for more information.

Prepare a Workload Replay

Provide steps to prepare the refreshable clone for a replay.

Perform the following steps to prepare for a workload replay:

- Refresh the refreshable clone to the capture start timestamp.

  You can find the capture start timestamp by querying the `DBA_WORKLOAD_CAPTURES` view. See DBA_WORKLOAD_CAPTURES for more information.
- Manually disconnect the refreshable clone.
- Optionally, you can also make changes to the refreshable clone. For example, changing parameter values, turning certain features on/off to see the impact on the replay.

> **Note:**
>
> This step is not applicable when you are replaying the workload on a full clone.

Replay a Workload on an Autonomous Database Instance

After you complete a workload capture, you can replay it on a test system. Oracle replays the actions recorded during workload capture, with the same timing, concurrency, and transaction dependencies of the production system.

Run `DBMS_CLOUD_ADMIN.REPLAY_WORKLOAD` to initiate workload replay on your database.

- To replay the captured workload:
  - You must be logged in as the `ADMIN` user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.
  - Provision a refreshable or full clone of the Autonomous Database instance on which you need to capture the workload.
  - If the replay target is a refreshable clone, you should refresh it to the capture start time and then disconnect it.

    You can retrieve the capture start time by querying the `START_TIME` column from the `DBA_WORKLOAD_CAPTURES` view on the Autonomous Database instance where the workload was captured. See DBA_WORKLOAD_CAPTURES for more information.
  - Replay the workload capture.

Example to replay a workload on a refreshable clone :

```
BEGIN
  DBMS_CLOUD_ADMIN.REPLAY_WORKLOAD(
      capture_name => 'CAP_TEST1');
END;
/
```

This example downloads the capture files from the Object Storage, replays the captured workload, and uploads a replay report after a replay.

The `CAPTURE_NAME` parameter specifies the name of the workload capture. This parameter is mandatory.

Example to replay a workload on a full clone :

```
BEGIN
  DBMS_CLOUD_ADMIN.REPLAY_WORKLOAD(
       capture_name                => 'CAP_TEST1',
       capture_source_tenancy_ocid => 'OCID1.TENANCY.REGION1..ID1',
       capture_source_db_name      => 'ADWFINANCE');
END;
/
```

> **Note:**
>
> If there are multiple captures with the same capture name, `REPLAY_WORKLOAD` uses the latest capture. Oracle recommends that you use a unique capture name for each capture to prevent confusion on which capture you are replaying.

This example downloads the capture files from the Object Storage, replays the captured workload, and uploads a replay report after a replay.

The `CAPTURE_NAME` parameter specifies the name of the workload capture. This parameter is mandatory.

The `CAPTURE_SOURCE_TENANCY_OCID` parameter specifies the source tenancy OCID of the workload capture. This parameter is mandatory when running the workload capture in a full clone.

The `CAPTURE_SOURCE_DB_NAME` parameter specifies the source database name of the workload capture. This parameter is mandatory when running the workload capture in a full clone.

You can query the `DBA_CAPTURE_REPLAY_STATUS` view to check the workload replay status.

See DBA_CAPTURE_REPLAY_STATUS View for more information.

> **Note:**
>
> You must subscribe to the Information event `com.oraclecloud.databaseservice.autonomous.database.information` to be notified about the start and completion of the `REPLAY_WORKLOAD`, as well as the Object Storage link to download the replay reports.
> The PAR URL is contained in the `replayDownloadURL` field of the event and is valid for 7 days from the date of generation. The PAR URL points to a zip file that contains a replay report in HTML and an AWR report. See Information Events on Autonomous Database for more information.

See REPLAY_WORKLOAD Procedure for more information.

Capture-Replay Workloads between non-Autonomous and Autonomous Databases

You can Capture and Replay from a non-Autonomous Database instance into an Autonomous Database.

This enables you to compare workloads between an on-prem database or other cloud service database and an Autonomous Database instance.

Capture-Replay workflow between non-Autonomous and Autonomous Databases consists of these steps:

- Capture the workload on the production system as described in:

  Capture a Workload

- Replay the captured workload, on the test system, as described in:

  Replay a Workload on an Autonomous Database Instance

- Capture a Workload
  The first step in using Database Replay is to capture the production workload.
- Replay a Workload on an Autonomous Database Instance
  After you complete a workload capture, you can replay it on a test system. Oracle replays on the test system the actions recorded during workload capture, with the same timing , concurrency, and transaction dependencies of the production system.

Capture a Workload

The first step in using Database Replay is to capture the production workload.

When you begin workload capture on the production system, all requests from external clients directed to Oracle Database are tracked and stored in binary files called capture files.

A workload capture results in the creation of two subdirectories, `cap` and `capfiles`, which contain the capture files.

The capture files provide all pertinent information about the client request, including transaction details, bind values, and SQL text.

These capture files are platform independent and can be transported to another system.

See Workload Capture to capture a workload on an on-premises database.

Replay a Workload on an Autonomous Database Instance

After you complete a workload capture, you can replay it on a test system. Oracle replays on the test system the actions recorded during workload capture, with the same timing , concurrency, and transaction dependencies of the production system.

Run `DBMS_CLOUD_ADMIN.REPLAY_WORKLOAD` to initiate workload replay on your database. You must be logged in as the `ADMIN` user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN` to run `REPLAY_WORKLOAD`.

Example to replay on an Autonomous Database instance a workload captured from an on-premises database:

```
BEGIN
   DBMS_CLOUD_ADMIN.REPLAY_WORKLOAD(
      location_uri    => 'https://objectstorage.us-
```

```
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
      credential_name => 'CRED_TEST',
      synchronization => TRUE,
      process_capture => TRUE);
END;
/
```

This downloads the capture files contained in the Object Storage location specified in the `location_uri` parameter, and replays the workload capture from the capture files. The replay generates and uploads the replay and Automatic Workload Repository reports to the Object Storage location specified in the `location_uri` parameter.

The `credential_name` parameter specifies the credential to access the object storage bucket. The credential that you supply must have the write privileges to write into the Object Storage bucket. The write privileges are required to upload the replay report to the bucket.

If you do not specify a `credential_name` value, then `DEFAULT_CREDENTIAL` is used.

The `synchronization` parameter specifies the synchronization method used during workload replay. A `TRUE` value indicates that the synchronization is SCN based.

The `process_capture` specifies whether you need to specify `process_capture` value or not. A `TRUE` value indicates that the replay includes `process_capture`.

> **Note:**
>
> You must maintain the same logical state of the capture and replay databases at the start of the capture time.

You can query the `DBA_CAPTURE_REPLAY_STATUS` view to check the workload replay status.

See DBA_CAPTURE_REPLAY_STATUS View for more information.

> **Note:**
>
> You must subscribe to the Information event `com.oraclecloud.databaseservice.autonomous.database.information` to be notified about the start and completion of the `REPLAY_WORKLOAD` as well as the Object Storage link to download the replay reports. This PAR URL is contained in the `replayDownloadURL` field of the event and is valid for 7 days from the date of generation. See Information Events on Autonomous Database for more information.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

See Navigate to Oracle Cloud Infrastructure Object Storage and Create Bucket for more information on Object Storage.

See Upload Files to Your Oracle Cloud Infrastructure Object Store Bucket for more information on uploading files to Object Storage.

You don't need to create a credential to access Oracle Cloud Infrastructure Object Store if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

See REPLAY_WORKLOAD Procedure for more information.

Test Workloads Against an Upcoming Patch

Using the workload auto replay feature you can automatically capture a workload from a production database that is on the regular patch level and replay the workload on a target refreshable clone that is on the early patch level. This feature allows you to test an upcoming patch by running an existing workload that is in production against a patch before the patch reaches production.

- About Testing Workloads Against an Upcoming Patch
  Using the workload auto replay feature you can automate the process of capture-replay to capture a workload that runs on a production database and automatically replay the workload on a target refreshable clone after an upcoming patch is applied on the target.
- Enable Workload Auto Replay
  The `WORKLOAD_AUTO_REPLAY` feature allows you run a workload from your production database and monitor for any divergence on an instance that is patched one week in advance. This feature allows you to test an upcoming patch by running an existing workload that is in production against a patch before the patch reaches production.
- Disable Workload Auto Replay
  Run `DBMS_CLOUD_ADMIN.DISABLE_FEATURE` to disable `WORKLOAD_AUTO_REPLAY`.

About Testing Workloads Against an Upcoming Patch

Using the workload auto replay feature you can automate the process of capture-replay to capture a workload that runs on a production database and automatically replay the workload on a target refreshable clone after an upcoming patch is applied on the target.

Autonomous Database provides the ability to provision an instance or create a refreshable clone with the **Early** patch level option. On instances running at the **Early** patch level, Autonomous Database applies upcoming maintenance patches a week before the patches are applied to production databases (databases that are provisioned at the **Regular** patch level). Using the `WORKLOAD_AUTO_REPLAY` feature you can assure that an upcoming patch is tested against your workload before the patch goes to production. This allows you to verify that the patch either fixes a known issue or does not introduce an issue that affects your workload.

To find information about captures and replays, subscribe to Information events. Information events provide notification for workload capture and reply events and include a PAR URL where you can download the capture file and replay report. See Subscribe to Information Events to be Notified of Capture and Replay Details for more information.

When `WORKLOAD_AUTO_REPLAY` is enabled the source database captures a workload by running for a specified number of minutes. By default the workload capture starts when you enable `WORKLOAD_AUTO_REPLAY` (optionally you can use parameters to set the capture start day and time). Next, Autonomous Database checks the target database to verify the patching status. After the upcoming weekly patch is applied, Autonomous Database replays the workload on the target database. This capture-replay cycle continues automatically each week with Autonomous Database capturing the workload on the source database, waiting for the upcoming patch to be applied, and replaying the workload on the refreshable clone.

Note the following for enabling `WORKLOAD_AUTO_REPLAY`:

- The source database must use the **Regular** patch level.

- The target database must use the **Early** patch level.
- The target database must be a refreshable clone of the source database, and must be created before you enable `WORKLOAD_AUTO_REPLAY`.
- A source database can enable `WORKLOAD_AUTO_REPLAY` for no more than one refreshable clone (you can enable this feature for a maximum of one refreshable clone, even if you create multiple refreshable clones from the same source database).
- After you enable `WORKLOAD_AUTO_REPLAY`, the capture-replay cycle continues every week. Autonomous Database runs a capture on the source database and then replays the workload on the target database, until you disable `WORKLOAD_AUTO_REPLAY`.

You can find information about workload captures and replays in the `DBA_CAPTURE_REPLAY_HISTORY` view. See DBA_CAPTURE_REPLAY_HISTORY View for more information.

Autonomous Database automatically applies patches on your database. Oracle provides a service level objective of zero regressions in your production database due to these patches. See Zero-Regression Service Level Objective for more information.

Enable Workload Auto Replay

The `WORKLOAD_AUTO_REPLAY` feature allows you run a workload from your production database and monitor for any divergence on an instance that is patched one week in advance. This feature allows you to test an upcoming patch by running an existing workload that is in production against a patch before the patch reaches production.

To enable `WORKLOAD_AUTO_REPLAY`:

1. Create a refreshable clone of the production database.

   When you create the target refreshable clone, set the patch level to **Early**.

   See Set the Patch Level and Create a Refreshable Clone for an Autonomous Database Instance for more information.

2. Run `DBMS_CLOUD_ADMIN.ENABLE_FEATURE` on the source database.

   For example:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_FEATURE(
        feature_name => 'WORKLOAD_AUTO_REPLAY',
        params       => JSON_OBJECT(
                            'target_db_ocid' VALUE
'OCID1.TENANCY.REGION..ID1',
                            'capture_duration' VALUE 120,
                            'capture_day' VALUE 'MONDAY',
                            'capture_time' VALUE '15:00'));
END;
/
```

   Where the parameters are:

   - `feature_name`: the value `WORKLOAD_AUTO_REPLAY` enables the workload auto replay feature.
   - `params`: is a JSON Object with the following value pairs:

- `target_db_ocid`: accepts a `string` value. The value specifies the OCID of the target refreshable clone database on which the captured workload is replayed.

  This parameter is mandatory.

- `capture_duration`: accepts a `number` value. The value specifies the duration in minutes for which the workload is captured on the production database. The value must be in the range between 1 and 720 minutes.

  This parameter is mandatory.

- `capture_day`: accepts a `string` value. The value specifies the day of the week the workload capture on the production database should begin.

  This parameter is optional.

- `capture_time`: accepts a value in the `HH24:MM` format. The value specifies the time of the day the workload capture on the production database should begin.

  This parameter is optional.

By default the workload capture starts when you enable `WORKLOAD_AUTO_REPLAY`. When the optional `capture_day` and `capture_time` are specified, the automatic workload capture and replay happen at the specified timestamp.

For example if `capture_day` is Monday and `capture_time` is 15:00, the first capture on the production database starts at 3PM on the next Monday. The same day of week and time are also used to schedule subsequent captures and replays.

See ENABLE_FEATURE Procedure for more information.

3. Query the `DBA_CAPTURE_REPLAY_STATUS` view to check the workload replay status.

   See Replay a Workload on an Autonomous Database Instance and DBA_CAPTURE_REPLAY_STATUS View for more information.

This example enables `WORKLOAD_AUTO_REPLAY` on the source Autonomous Database and on the specified target refreshable clone database. With `WORKLOAD_AUTO_REPLAY` enabled, every week Autonomous Database runs a capture on the source database and replays the workload on the target database, until you disable `WORKLOAD_AUTO_REPLAY`.

To find information about captures and replays, subscribe to Information events. Information events provide notification for workload capture and reply events and include a PAR URL where you can download the capture file and replay report. See Subscribe to Information Events to be Notified of Capture and Replay Details for more information.

You can find information about workload captures and replays in the `DBA_CAPTURE_REPLAY_HISTORY` view. See DBA_CAPTURE_REPLAY_HISTORY View for more information.

Disable Workload Auto Replay

Run `DBMS_CLOUD_ADMIN.DISABLE_FEATURE` to disable `WORKLOAD_AUTO_REPLAY`.

Run `DBMS_CLOUD_ADMIN.DISABLE_FEATURE` to disable workload auto replay. For example:

```
BEGIN
DBMS_CLOUD_ADMIN.DISABLE_FEATURE(
    feature_name => 'WORKLOAD_AUTO_REPLAY');
END;
/
```

You must be logged in as ADMIN or have DBMS_CLOUD_ADMIN privileges to run DBMS_CLOUD_ADMIN.DISABLE_FEATURE.

See DISABLE_FEATURE Procedure for more information.

Manage and Store Files in a Cloud Code Repository with Autonomous Database

Autonomous Database provides routines to manage and store files in Cloud Code (Git) Repositories. The supported Cloud Code Repositories are: GitHub, AWS CodeCommit, and Azure Repos.

- About Cloud Code Repositories with Autonomous Database
  The DBMS_CLOUD_REPO package provides a single interface for accessing a Cloud Code Repository from Autonomous Database.
- Initialize a Cloud Code Repository
  The DBMS_CLOUD_REPO initialization routines initialize a Cloud Code Repository. After you obtain a Cloud Code Repository handle, you use the handle to access the Cloud Code Repository.
- Create and Manage a Cloud Code Repository
  The DBMS_CLOUD_REPO management routines allow you to manage a Cloud Code Repository by creating, listing, updating, or deleting a repository.
- Create and Manage Branches in a Cloud Code Repository
  The DBMS_CLOUD_REPO management routines allow you to manage Cloud Code Repository branches by creating, listing, merging, or deleting branches in a repository.
- Export Schema Objects to the Cloud Code Repository Branch
  The DBMS_CLOUD_REPO management routine allows you to export metadata of the objects in a schema to the Cloud Code Repository branch. You can filter your list based on the object names or object types.
- Use File Operations with a Cloud Code Repository
  The DBMS_CLOUD_REPO file operations allow you to create, get, list, update, or delete files in a Cloud Code Repository.
- Use SQL Install Operations with a Cloud Code Repository
  The DBMS_CLOUD_REPO SQL Install operations allow you to store and download SQL scripts from a Cloud Code Repository.

About Cloud Code Repositories with Autonomous Database

The DBMS_CLOUD_REPO package provides a single interface for accessing a Cloud Code Repository from Autonomous Database.

The supported Cloud Code Repositories provide the following features:

- Git Version Control System: Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows.
- Git Repository: A Git repository is a virtual storage of your project. It allows you to save versions of your code, which you can access when needed.

The DBMS_CLOUD_REPO APIs use a repository handle (REPO object). The repository handle is an opaque JSON object that represents a Cloud Code Repository of a specific cloud provider. A

REPO object can be passed to different DBMS_CLOUD_REPO APIs. This opaque object ensures that DBMS_CLOUD_REPO procedures and functions are multicloud compatible; you do not have to change your code when you migrate from one Cloud Code Repository provider to another Cloud Code Repository.

Autonomous Database provides the following to help you work with Cloud Code Repositories:

- Repository initialization operations to allow you to initialize a repository.

  See Initialize a Cloud Code Repository for more information.
- Repository management operations that let you create, list, update or delete a repository.

  See Create and Manage a Cloud Code Repository for more information.
- Repository branch management operations that let you create, list, merge or delete branches in a repository.

  See Create and Manage Branches in a Cloud Code Repository for more information.
- Export the metadata DDL of all objects in a schema to a repository.

  See Export Schema Objects to the Cloud Code Repository Branch for more information.
- Repository file management operations to upload, download, update, and delete files.

  See Use File Operations with a Cloud Code Repository for more information.
- SQL install operations that let you export database object metadata DDL to a repository and install SQL statements into the database from a Cloud Code Repository.

  See Use SQL Install Operations with a Cloud Code Repository for more information.

Initialize a Cloud Code Repository

The DBMS_CLOUD_REPO initialization routines initialize a Cloud Code Repository. After you obtain a Cloud Code Repository handle, you use the handle to access the Cloud Code Repository.

To initialize a Cloud Code Repository:

1. Create a credential to access the Cloud Code Repository.

   See CREATE_CREDENTIAL Procedure for information on creating credentials.
2. Depending on the repository, GitHub, Azure Repos, or AWS CodeCommit, call DBMS_CLOUD_REPO.INIT_REPO with the parameters for the particular repository to obtain a repository handle.

   The following examples provide samples for each supported Cloud Code Repository.

   - **GitHub Initialization:**

```
DEFINE repo_name='test_repo';
DEFINE cred_name='GITHUB_CRED';
VAR repo clob
BEGIN
  :repo := DBMS_CLOUD_REPO.INIT_REPO(
               params => JSON_OBJECT('provider' value 'github',
                                     'repo_name' value '&repo_name',
                                     'credential_name' value
'&cred_name',
                                     'owner'           value
'<myuser>')
          );
```

```
END;
/
```

- **Azure Repos Initialization:**

```
DEFINE repo_name='test_repo';
DEFINE cred_name='AZURE_REPO_CRED';
VAR repo clob
BEGIN
  :repo := DBMS_CLOUD_REPO.INIT_REPO(
               params => JSON_OBJECT('provider' value 'azure',
                                     'repo_name' value '&repo_name',
                                     'credential_name' value
'&cred_name',
                                     'organization' value '<myorg>',
                                     'project' value '<myproject>')
          );
END;
/
```

- **AWS CodeCommit Initialization:**

```
DEFINE repo_name='test_repo';
DEFINE cred_name='AWS_REPO_CRED';
VAR repo clob
BEGIN
  :repo := DBMS_CLOUD_REPO.INIT_REPO(
               params => JSON_OBJECT('provider' value 'aws',
                                     'repo_name' value '&repo_name',
                                     'credential_name' value
'&cred_name',
                                     'region' value 'us-east-1')
          );
END;
/
```

See DBMS_CLOUD_REPO Initialization Operations for details on the initialization functions.

Create and Manage a Cloud Code Repository

The `DBMS_CLOUD_REPO` management routines allow you to manage a Cloud Code Repository by creating, listing, updating, or deleting a repository.

First, obtain a Cloud Code Repository handle to provide access a repository. See Initialize a Cloud Code Repository for details.

1. To create a repository:

```
VAR repo clob
BEGIN
  DBMS_CLOUD_REPO.CREATE_REPOSITORY(
        repo => :repo,
        description => 'test repo'
  );
```

```
END;
/
```

2. To update a repository:

```
VAR repo clob
DEFINE repo_name='test_repo';
BEGIN
  DBMS_CLOUD_REPO.UPDATE_REPOSITORY(
        repo => :repo,
        new_name => '&repo_name' || '_new'
  );
END;
/
```

3. To list repositories:

```
col id format a30
col name format a10
col description format a15
select id, name, bytes, private, description from
  DBMS_CLOUD_REPO.LIST_REPOSITORIES(:repo);
```

4. To delete a repository:

```
VAR repo clob
BEGIN
  DBMS_CLOUD_REPO.DELETE_REPOSITORY(
        repo => :repo
  );
END;
/
```

See DBMS_CLOUD_REPO Repository Management Operations for more information.

Create and Manage Branches in a Cloud Code Repository

The `DBMS_CLOUD_REPO` management routines allow you to manage Cloud Code Repository branches by creating, listing, merging, or deleting branches in a repository.

To perform Cloud Code Repository branch management operations you must first:

- Create a credential.

  See CREATE_CREDENTIAL Procedure for details.
- Obtain a handle.

  See Initialize a Cloud Code Repository for details.
- Create a repository.

  See Create and Manage a Cloud Code Repository for details.
- Log in as the ADMIN user or have `EXECUTE` privilege on `DBMS_CLOUD_REPO`.

1. To create a branch in a Cloud Code Repository:

```
BEGIN
  DBMS_CLOUD_REPO.CREATE_BRANCH (
    repo            => l_repo,
    branch_name     => 'test_branch',
    parent_branch   => 'main'
    );
END;
/
```

2. To delete a branch in a Cloud Code Repository:

```
BEGIN
  DBMS_CLOUD_REPO.DELETE_BRANCH (
      repo        => l_repo,
      branch_name => 'test_branch'
  );
END;
/
```

3. To merge a branch into another branch in a Cloud Code Repository:

```
BEGIN
  DBMS_CLOUD_REPO.MERGE_BRANCH (
    repo               => l_repo,
    branch_name        => 'test_branch',
    target_branch_name => 'main'
    );
END;
/
```

4. To list branches in a Cloud Code Repository:

```
SELECT * FROM DBMS_CLOUD_REPO.LIST_BRANCHES (repo => l_repo);
```

5. You can list commits in a branch in Cloud Code Repository based on the following;
   - Repository
   - Branch
   - The file path of the branch
   - SHA/commit_id

> **Note:**
>
> SHA is a checksum 40-character string composed of hexadecimal characters (0–9 and a–f).

To list commits based on the repository:

```
SELECT * FROM DBMS_CLOUD_REPO.LIST_COMMITS (
    repo => :repo);
```

To list commits based on the repository, commit_id and file path of the branch:

```
SELECT * FROM DBMS_CLOUD_REPO.LIST_COMMITS (
    repo        => l_repo,
    commit_id   => '66dd2b23b74cd0afabd11af66c6aa9c550540ba6',
    file_path   => 'sub_dir/test11.sql'
```

To list commits based on the repository, and file path of the branch:

```
SELECT * FROM DBMS_CLOUD_REPO.LIST_COMMITS (
    repo        => :repo,
    branch_name => 'branch1');
```

To list commits based on the repository, branch_name and file path of the branch:

```
SELECT * FROM DBMS_CLOUD_REPO.LIST_COMMITS (
    repo         => :repo,
    branch_name  => 'branch1',
    file_path    => 'sub_dir/test11.sql');
```

To list commits based on the repository and commit_id of the branch:

```
SELECT * FROM DBMS_CLOUD_REPO.LIST_COMMITS (
    repo => :repo,
    commit_id =>'66dd2b23b74cd0afabd11af66c6aa9c550540ba6');
```

See DBMS_CLOUD_REPO Repository Branch Management Operations for more information.

Export Schema Objects to the Cloud Code Repository Branch

The `DBMS_CLOUD_REPO` management routine allows you to export metadata of the objects in a schema to the Cloud Code Repository branch. You can filter your list based on the object names or object types.

To export schema metadata you must first:

- Create a credential.

  See CREATE_CREDENTIAL Procedure for details.
- Obtain a handle.

  See Initialize a Cloud Code Repository for details.
- Create a repository.

  See Create and Manage a Cloud Code Repository for details.
- Log in as the ADMIN user or have `EXECUTE` privilege on `DBMS_CLOUD_REPO`.

Use the `EXPORT_SCHEMA` procedure to export metadata of the objects in your schema to a Cloud Code Repository branch:

```
BEGIN
  DBMS_CLOUD_REPO.EXPORT_SCHEMA(
    repo          => l_repo,
    schema_name   => 'USER1',
    file_path     => 'myschema_ddl.sql'
    filter_list   =>
        to_clob('[
             { "match_type":"equal",
                 "type":"table"
             },
             { "match_type":"not_equal",
                 "type":"view"
             },
             { "match_type":"in",
                 "type":"table",
                 "name": " ''EMPLOYEE_SALARY'',''EMPLOYEE_ADDRESS'' "
             },
             { "match_type":"equal",
                 "type":"sequence",
                 "name": "EMPLOYEE_RECORD_SEQ"
             },
             { "match_type":"like",
                 "type":"table",
                 "name": "%OFFICE%"
             }
        ]'
     );
  );
END;
/
```

This example exports the metadata of the `USER1` schema into the `l_repo` repository. The export includes the metadata of the tables `EMPLOYEE_SALARY` and `EMPLOYEE_ADDRESS`, and any table name that contains `OFFICE`. It also exports the `EMPLOYEE_RECORD_SEQ` sequence and excludes the views in the schema.

See EXPORT_SCHEMA Procedure for more information.

Use File Operations with a Cloud Code Repository

The `DBMS_CLOUD_REPO` file operations allow you to create, get, list, update, or delete files in a Cloud Code Repository.

Obtain a Cloud Code Repository handle before using the file operations. See Initialize a Cloud Code Repository for details.

You also need to create a repository before you work with files. See Create and Manage a Cloud Code Repository for details.

1. To get a file:

```
SELECT DBMS_CLOUD_REPO.GET_FILE(repo => :repo, file_path => 'test1.sql')
```

2. To create a file:

```
VAR repo clob
BEGIN
  DBMS_CLOUD_REPO.PUT_FILE(
        repo => :repo,
        file_path => 'test1.sql',
        contents => UTL_RAW.cast_to_raw('create table t1 (x varchar2(30))'
|| CHR(10) || '/')
  );
END;
/
```

3. To update a file:

```
VAR repo clob
BEGIN
  DBMS_CLOUD_REPO.PUT_FILE(
        repo => :repo,
        file_path => 'test1.sql',
        contents => UTL_RAW.cast_to_raw('create table t2 (x varchar2(30))'
|| CHR(10) || '/')
  );
END;
/
```

4. To list files:

```
SELECT id, name, bytes, url FROM DBMS_CLOUD_REPO.LIST_FILES(repo => :repo);
```

5. To delete a file:

```
VAR repo clob
BEGIN
  DBMS_CLOUD_REPO.DELETE_FILE(
        repo => :repo,
        file_path => 'test1.sql'
  );
END;
/
```

See DBMS_CLOUD_REPO File Operations for more information.

Use SQL Install Operations with a Cloud Code Repository

The `DBMS_CLOUD_REPO` SQL Install operations allow you to store and download SQL scripts from a Cloud Code Repository.

Obtain a Cloud Code Repository handle before using the SQL Install operations. See Initialize a Cloud Code Repository for details.

You also need to create a repository before you work with SQL Install operations. See Create and Manage a Cloud Code Repository for details.

The scripts are intended as schema install scripts and not as generic SQL scripts:

- Scripts cannot contain SQL*Plus client specific commands.
- Scripts cannot contain bind variables or parameterized scripts.
- SQL statements must be terminated with a slash on a new line (/).
- Scripts can contain DDL, DML PLSQL statements, but direct `SELECT` statements are not supported. Using SELECT within a PL/SQL block is supported.

Any SQL statement that can be run using `EXECUTE IMMEDIATE` will work if it does not contain bind variables or defines.

1. To upload DDL metadata to a Cloud Code Repository:

```
VAR repo clob
BEGIN
  DBMS_CLOUD_REPO.EXPORT_OBJECT(
        repo => :repo,
        object_type => 'PACKAGE',
        object_name => 'MYPACK',
        file_path   => 'mypack.sql'
  );
END;
/
```

2. To install SQL statements from a file:

```
VAR repo clob
BEGIN
  DBMS_CLOUD_REPO.INSTALL_FILE(
        repo => :repo,
        file_path     => 'test3.sql',
        stop_on_error => FALSE
  );
END;
/
```

3. To install SQL statements from a buffer:

```
VAR repo clob
BEGIN
  DBMS_CLOUD_REPO.INSTALL_SQL(
        repo => :repo,
        content   => 'create table t1 (x varchar2(30))' || CHR(10) || '/',
        stop_on_error => FALSE
  );
END;
/
```

See DBMS_CLOUD_REPO SQL Install Operations for more information.

Invoke User Defined Functions

You can invoke external functions such as OCI, AWS Lambda and Azure remote functions in your Autonomous Database as SQL functions.

- About User Defined Functions
  User defined functions enables you to invoke an external available function from PL/SQL or SQL code within your database. You can invoke Oracle Cloud Infrastructure Functions, AWS Lambda, Azure Functions and External procedures using user defined functions.
- Steps to Invoke OCI Cloud Functions as SQL Functions
  Shows the steps to invoke OCI remote functions as SQL functions in your database.
- Steps to Invoke AWS Lambda as SQL Functions
  Shows the steps to invoke AWS remote functions as SQL functions in your database.
- Steps to Invoke Azure Function as SQL Functions
  Shows the steps to invoke Azure remote functions as SQL functions in your database using HTTP trigger.
- Invoke External Procedures as SQL Functions
  Shows the steps to invoke external procedures using PL/SQL within your database.

About User Defined Functions

User defined functions enables you to invoke an external available function from PL/SQL or SQL code within your database. You can invoke Oracle Cloud Infrastructure Functions, AWS Lambda, Azure Functions and External procedures using user defined functions.

Oracle Cloud Infrastructure Functions are fully managed, multi-tenant, highly scalable, on-demand, Functions-as-a-Service platform. Oracle Cloud Infrastructure Functions are built on enterprise-grade Oracle Cloud Infrastructure and powered by the Fn Project open-source engine. Use Oracle Cloud Infrastructure Functions also referred to as OCI Functions, to focus on developing code to meet the business requirements.

See Overview of Functions for more information.

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any application or backend service without provisioning or managing servers.

See AWS Lambda for more information.

Azure Function is a serverless compute service that enables users to run event-triggered code without provisioning or managing infrastructure.

Please review the Azure documentation at the links below:

- Azure Functions overview
- Create your first function in the Azure portal

External procedures are functions written in a third-generation language (C, for example) and callable from within PL/SQL or SQL as if they were a PL/SQL routine or function.

See What Is an External Procedure? for more information.

Steps to Invoke OCI Cloud Functions as SQL Functions

Shows the steps to invoke OCI remote functions as SQL functions in your database.

To invoke cloud functions in OCI as SQL functions, we will be creating a catalog of SQL wrapper functions that reference and call their respective cloud function via their API endpoints. Before you create this catalog, it is assumed here that you have created the necessary cloud functions to be referenced by this catalog.

See Creating and Deploying Functions for more information on the creation and deployment of Oracle Cloud Infrastructure Functions and Application (ie a group of OCI Functions).

1. Create credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`. The credential object that you are creating must be of type private key.

```
SET DEFINE OFF
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL (
       credential_name => 'OCI_CRED',
       user_ocid       => 'user_ocid',
       tenancy_ocid    => 'tenancy_ocid',
       private_key     => 'private_key',
       fingerprint     => 'fingerprint'
  );
END;
/

PL/SQL procedure successfully completed.
```

   This creates the `OCI_CRED` credential.

   See CREATE_CREDENTIAL Procedure for more information.

   > **Note:**
   >
   > If you are using a Resource Principal for authentication, the necessary policies required for OCI Function access must be configured. See Details for Functions for more information.

2. Create a catalog.

   A catalog is a collection of wrapper functions that reference and call their respective cloud functions via their API endpoints.

   Example to create a catalog for Oracle Cloud Infrastructure Functions.

```
BEGIN
DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
    credential_name  => 'OCI_CRED',
    catalog_name     => 'OCI_DEMO_CATALOG',
    service_provider => 'OCI',
    cloud_params     => '{"region_id":"phx",
      "compartment_id":"compartment_id"}'
);
END;
```

```
/

PL/SQL procedure successfully completed.
```

This creates the `OCI_DEMO_CATALOG` catalog.

See CREATE_CATALOG Procedure for more information. You can query DBA_CLOUD_FUNCTION_CATALOG View and USER_CLOUD_FUNCTION_CATALOG View views to retrieve the list of all the catalogs in your database.

3. You can list the functions in a catalog.

   Following is the example to list Oracle Cloud Infrastructure Functions:

```
VAR function_list CLOB;

BEGIN
    DBMS_CLOUD_FUNCTION.LIST_FUNCTIONS (
      credential_name  => 'OCI_CRED',
      catalog_name     => 'OCI_DEMO_CATALOG',
      function_list    => :function_list
 );
END;
/

PL/SQL procedure successfully completed.

SELECT JSON_QUERY (:function_list, '$' RETURNING VARCHAR2(32676) pretty)
AS search_results FROM dual;
SEARCH_RESULTS                          --This is a sample output
--------------------------------------------------------------------------------
--------------------
[
  {
    "functionName"   : "create_par",
    "functionId"     : "ocid.funfc.oc1.phx.aaaa_example",
    "invokeEndpoint" : "https://dw.us.func.oci.oraclecloud_example.com"
  },
  {
    "functionName"   : "fintech",
    "functionId"     : "ocid.funfc.oc1.phx.bbbb_example"
SEARCH_RESULTS
--------------------------------------------------------------------------------
--------------------
4ayosyosv4sthmya2lyza",
    "invokeEndpoint" : "https://dw.us.func.oci.oraclecloud.com_example"
  },
  {
    "functionName"   : "jwt_codec",
    "functionId"     : "ocid.funfc.oc1.phx.jwt_code_example",
    "invokeEndpoint" : "https://dw.us.func.oci.oraclecloud_example.com"
  },
SEARCH_RESULTS
--------------------------------------------------------------------------------
--------------------
  {
    "functionName"   : "oci-objectstorage-create-par-python",
```

```
    "functionId"     : "ocid.funfc.oc1.phx.aaaaaaaas_example",
    "invokeEndpoint" : "https://dw.us.func.oci.oraclecloud_example.com"
  },
  {
    "functionName"   : "run_dbt",
    "functionId"     : "ocid.funfc.oc1.phx.aaaaaaaav_example",

SEARCH_RESULTS
------------------------------------------------------------------------
----------------------
    "invokeEndpoint" : "https://dw.us.func.oci.oraclecloud_example.com"
  }
]
```

See LIST_FUNCTIONS Procedure for more information.

4. Run the `DBMS_CLOUD_FUNCTION.SYNC_FUNCTIONS` to create wrapper SQL functions. You can use one of the following methods to create the wrapper SQL functions in the catalog, that call their respective cloud functions:

   - **SYNC_FUNCTIONS**: `SYNC_FUNCTIONS` is the quickest and simplest method, which automatically syncs (creates or deletes) wrapper functions in the catalog with the complete list of cloud functions defined in the region, compartment, and tenancy with which the catalog was created. For example:

```
BEGIN
 DBMS_CLOUD_FUNCTION.SYNC_FUNCTIONS (
        catalog_name => 'OCI_DEMO_CATALOG'
 );
END;
/
PL/SQL procedure successfully completed.
```

This creates a PL/SQL wrapper for adding new functions to the catalog and removing wrappers for functions that have been deleted from the catalog.

Run the following query to verify the sync.

```
SELECT object_name FROM sys.all_objects WHERE owner='TEST_USER' AND
object_type='FUNCTION';

OBJECT_NAME
------------------------------------------------------------------------
-----
CREATE_PAR
FINTECH
JWT_CODEC
OCI-OBJECTSTORAGE-CREATE-PAR-PYTHON
RUN_DBT
```

> **Note:**
>
> Keep a note of the current user in order to run this command.

See SYNC_FUNCTIONS Procedure for more information.

- You can manually create a `SQL` Function in your catalog that calls its respective cloud function using `DBMS_CLOUD.CREATE_FUNCTION`. For example:

Example to create a function in the `OCI_DEMO_CATALOG` catalog.

```
VAR function_args CLOB;
EXEC :function_args := TO_CLOB('{"command": "VARCHAR2", "value":
"VARCHAR2"}');
BEGIN
    DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
        credential_name  => 'OCI_CRED',
        catalog_name     => 'OCI_DEMO_CATALOG',
        function_name    => 'fintech_fun',
        function_id      => 'ocid1.fnfunc.oc1.phx.aaabbbcccc_example',
        input_args       => :function_args
 );
END;
/

PL/SQL procedure successfully completed.
```

This creates the `FINTECH_FUN` function in the `OCI_DEMO_CATALOG` catalog.

The `FINTECH_FUN` function in the catalog is a reference to the respective cloud function whose endpoint is referenced by the `FUNCTION_ID` parameter. Invoking the function in the catalog along with its arguments runs the corresponding cloud function and provides the output returned by the cloud function.

Manually creating a function in the catalog also allows you to create custom return types and response handlers. For example:

```
CREATE OR REPLACE TYPE fintech_rt AS OBJECT (
    STATUS  VARCHAR2(1000),
    OUTPUT  CLOB
);
/

Type created.


CREATE OR REPLACE FUNCTION fintech_response_handler(function_response in
CLOB)
RETURN fintech_rt
IS
l_comp   fintech_rt;
l_json_obj  JSON_OBJECT_T;
status  VARCHAR2(1000);
output  CLOB;
BEGIN
```

```
      l_json_obj := JSON_OBJECT_T.parse(function_response);
      status     := l_json_obj.get('STATUS').to_string;
      output     := l_json_obj.get('RESPONSE_BODY').to_string;
      l_comp     := fintech_rt(status,output);
      RETURN l_comp;
END;
/
Function created.

VAR input_param clob;
VAR  l_return_type varchar2(100);
VAR  l_reponse_handler varchar2(1000);

exec :input_param       := TO_CLOB('{"command": "VARCHAR2", "value":
"VARCHAR2"}');
exec :l_return_type     := 'fintech_rt';
exec :l_reponse_handler := 'fintech_response_handler';

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (CREDENTIAL_NAME => 'OCI_CRED',
FUNCTION_NAME => 'fintech_fun', CATALOG_NAME => 'OCI_DEMO_CATALOG',
FUNCTION_ID => 'ocid1.funfn.oci.phx.aaaaaa_example', INPUT_ARGS
=> :input_param, RETURN_TYPE => :l_return_type ,REPONSE_HANDLER
=>:l_reponse_handler);

PL/SQL procedure successfully completed.
```

You can query DBA_CLOUD_FUNCTION View and USER_CLOUD_FUNCTION View views to retrieve the list of all the functions in your database.

See CREATE_FUNCTION Procedure for more information.

5. After the function is created you can `DESCRIBE` and invoke it.

```
DESC fintech_fun
COLUMN STATUS format a30
COLUMN OUTPUT format a30


DECLARE
l_comp fintech_rt;
BEGIN
l_comp := fintech_fun(command=>'tokenize',value => 'PHI_INFORMATION');
DBMS_OUTPUT.put_line ('Status of the function   =  '|| l_comp.status);
DBMS_OUTPUT.put_line ('Response of the function =  '|| l_comp.output);
END;
/

PL/SQL procedure successfully completed.
```

This invokes the `fintech_fun` cloud function by calling the function reference `oocid1.funfn.oci.phx.aaaaaa_example` in the `OCI_DEMO_CATALOG` catalog.

6. You can drop an existing function using `DROP_FUNCTION` procedure. For example:

```
EXEC DBMS_CLOUD_FUNCTION.DROP_FUNCTION (CATALOG_NAME =>
'OCI_DEMO_CATALOG', FUNCTION_NAME => 'fintech_fun');

PL/SQL procedure successfully completed.
```

This drops the `FINTECH_FUN` function from the `OCI_DEMO_CATALOG` catalog.

See DROP_FUNCTION Procedure for more information.

7. You can drop an existing catalog using `DROP_CATALOG` procedure. For example:

```
BEGIN
    DBMS_CLOUD_FUNCTION.DROP_CATALOG (
      catalog_name     => 'OCI_DEMO_CATALOG'
  );
END;
/

PL/SQL procedure successfully completed.
```

This drops the `OCI_DEMO_CATALOG` from your database.

See DROP_CATALOG Procedure for more information.

Steps to Invoke AWS Lambda as SQL Functions

Shows the steps to invoke AWS remote functions as SQL functions in your database.

To invoke AWS Lambda as SQL functions, we will be creating a catalog of SQL wrapper functions that reference and call their respective cloud function via their API endpoints. Before you create this catalog, it is assumed here that you have created the necessary cloud functions to be referenced by this catalog.

> **Note:**
>
> To access AWS lambda functions you need to configure the necessary policies. See Creating an IAM policy to access AWS Lambda resources and Using resource-based policies for Lambda for more information.

1. Create a credential using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`. The credential object that you are creating must be of type secret key.

```
SET DEFINE OFF
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name => 'AWS_CRED',
    username        => 'access_key_ID',
    password        => 'secret_access_key'
  );
```

```
END;
/
```

This creates the `AWS_CRED` credential.

See CREATE_CREDENTIAL Procedure for more information.

2. Create a catalog.

   A catalog is a collection of wrapper functions that reference and call their respective cloud functions via their API endpoints.

   Example to create a catalog for AWS functions.

```
BEGIN
    DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
        credential_name  => 'AWS_CRED',
        catalog_name     => 'AWS_DEMO_CATALOG',
        service_provider => 'AWS',
        cloud_params     => '{"region_id":"ap-northeast-1"}'
 );
END;
/

PL/SQL procedure successfully completed.
```

   This creates the `AWS_DEMO_CATALOG` catalog.

   See CREATE_CATALOG Procedure for more information. You can query DBA_CLOUD_FUNCTION_CATALOG View and USER_CLOUD_FUNCTION_CATALOG View views to retrieve the list of all the catalogs in your database.

3. You can list the functions in a catalog.

   Following is the example to list AWS Lambda functions:

```
VAR function_list CLOB;

BEGIN
 DBMS_CLOUD_FUNCTION.LIST_FUNCTIONS (
    credential_name  => 'AWS_CRED',
    catalog_name     => 'AWS_DEMO_CATALOG',
    function_list    => :function_list
);
END;
/

PL/SQL procedure successfully completed.

SELECT JSON_QUERY (:function_list, '$' RETURNING VARCHAR2(32676) pretty)
AS search_results FROM dual;

SEARCH_RESULTS                          --This is a sample output
--------------------------------------------------------------------------------
----------------------
[
  {
    "functionName"   : "test3_example",
```

```
    "functionArn"    : "arn:aws:lambda:ap-north-1:378:func:test3_example",
    "invokeEndpoint" : "https://swiy3.lambda-url.ap-
north-1.on.aws_example/"
  },
  {
    "functionName"   : "SumOfNum_example",
    "functionArn"    : "arn:aws:lambda:ap-
north-1:378:func:SumOfNum_example"

SEARCH_RESULTS
------------------------------------------------------------------------------
------------------------
    "invokeEndpoint" : "https://swiy3.lambda-url.ap-
north-1.on.aws_example/"
  },
  {
    "functionName"   : "testlambda_example",
    "functionArn"    : "arn:aws:lambda:ap-
north-1:378:func:testlambda_example",
    "invokeEndpoint" : "https://swiy3.lambda-url.ap-
north-1.on.aws_example/"
  },

SEARCH_RESULTS
------------------------------------------------------------------------------
------------------------
  {
    "functionName"   : "hellp-python_example",
    "functionArn"    : "arn:aws:lambda:ap-north-1:378:func:hellp-
python_example",
    "invokeEndpoint" : "https://swiy3.lambda-url.ap-
north-1.on.aws_example/"
  },
  {
    "functionName"   : "testlam_example",
    "functionArn"    : "arn:aws:lambda:ap-
north-1:378:func:testlam_example",

SEARCH_RESULTS
------------------------------------------------------------------------------
------------------------
    "invokeEndpoint" : "https://swiy3.lambda-url.ap-
north-1.on.aws_example/"
  }
]
```

See LIST_FUNCTIONS Procedure for more information.

4. Run the `DBMS_CLOUD_FUNCTION.SYNC_FUNCTIONS` to create wrapper SQL functions. You can use one of the following methods to create the wrapper SQL functions in the catalog, that call their respective cloud functions:

   - **SYNC_FUNCTIONS**: `SYNC_FUNCTIONS` is the quickest and simplest method, which automatically syncs (creates or deletes) wrapper functions in the catalog with the

complete list of cloud functions defined in the region, compartment, and tenancy with which the catalog was created. For example:

```
BEGIN
 DBMS_CLOUD_FUNCTION.SYNC_FUNCTIONS (
        catalog_name => 'AWS_DEMO_CATALOG'
 );
END;
/
PL/SQL procedure successfully completed.
```

This creates a PL/SQL wrapper for adding new functions to the catalog and removing wrappers for functions that have been deleted from the catalog.

Run the following query to verify the sync.

```
SELECT object_name FROM sys.all_objects WHERE owner='TEST_USER' AND
object_type='FUNCTION';


OBJECT_NAME
--------------------------------------------------------------------------------
-----
TESTLAMBDA
HELLP-PYTHON
TESTLAM
TEST3
SUMOFNUMBERS
```

> **Note:**
>
> Keep a note of the current user in order to run this command.

See SYNC_FUNCTIONS Procedure for more information.

- You can manually create a `SQL` Function in your catalog that calls its respective cloud function using `DBMS_CLOUD.CREATE_FUNCTION`.

Example to create a function in the `AWS_DEMO_CATALOG` catalog.

```
BEGIN
DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
  credential_name => 'AWS_CRED',
  catalog_name    => 'AWS_DEMO_CATALOG',
  function_name   => 'aws_testlambda',
  function_id     => 'arn:aws:lambda:ap-
northeast-1:378079562280:function:hellp-python'
);
END;
/
PL/SQL procedure successfully completed.
```

This creates the AWS_TESTLAMBDA function in the AWS_DEMO_CATALOG catalog.

The AWS_TESTLAMBDA function in the catalog is a reference to the respective cloud function whose endpoint is referenced by the FUNCTION_ID parameter. Invoking the function in the catalog along with its arguments runs the corresponding cloud function and provides the output returned by the cloud function.

You can query DBA_CLOUD_FUNCTION View and USER_CLOUD_FUNCTION View views to retrieve the list of all the functions in your database.

See CREATE_FUNCTION Procedure for more information.

5. After the function is created you can DESCRIBE and invoke it.

```
DESC AWS_TESTLAMBDA
COLUMN STATUS format a30
COLUMN OUTPUT format a30


SELECT AWS_TESTLAMBDA(NULL) FROM dual;

AWS_TESTLAMBDA(NULL)
----------------------------------------------------
{"STATUS":"200","RESPONSE_BODY":"Hello Python!!"}
```

This invokes the AWS_TESTLAMBDA function by calling the function reference arn:aws:lambda:ap-northeast-1:378079562280:function:hellp-python in the AWS_DEMO_CATALOG catalog.

6. You can drop an existing function using DROP_FUNCTION procedure. For example:

```
EXEC DBMS_CLOUD_FUNCTION.DROP_FUNCTION (CATALOG_NAME =>
'AWS_DEMO_CATALOG', FUNCTION_NAME => 'AWS_TESTLAMBDA');

PL/SQL procedure successfully completed.
```

This drops the AWS_TESTLAMBDA function from the AWS_DEMO_CATALOG catalog.

See DROP_FUNCTION Procedure for more information.

7. You can drop an existing catalog using DROP_CATALOG procedure. For example:

```
BEGIN
    DBMS_CLOUD_FUNCTION.DROP_CATALOG (
      catalog_name     => 'AWS_DEMO_CATALOG'
  );
END;
/

PL/SQL procedure successfully completed.
```

This drops the AWS_DEMO_CATALOG from your database.

See DROP_CATALOG Procedure for more information.

Steps to Invoke Azure Function as SQL Functions

Shows the steps to invoke Azure remote functions as SQL functions in your database using HTTP trigger.

To invoke Azure Function as SQL functions, create a catalog of SQL wrapper functions that reference and call their respective cloud function through their API endpoints. Before you create this catalog, it is assumed here that you have created the necessary Azure functions to be referenced by this catalog.

1. To access Azure functions you need to use Azure Service Principal with Autonomous Database. You must grant the **Website Contributor** role to the Azure Service Principal for the Azure function app under its **Access control** (IAM).

   See the following for more information:

   - Use Azure Service Principal to Access Azure Resources
   - Azure built-in roles
   - Assign Azure roles using the Azure portal

2. Create a catalog.

   A catalog is a collection of wrapper functions that reference and call their respective cloud functions via their API endpoints.

   Example to create a catalog for Azure functions.

```
BEGIN
    DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
        credential_name  => 'AZURE$PA',
        catalog_name     => 'AZURE_DEMO_CATALOG',
        service_provider => 'AZURE',
        cloud_params     =>
'{"subscription_id":"XXXXXXXXXXXXXXXXXXXXXXXX_example"}'
 );
END;
/
```

   The `SUBSCRIPTION_ID` value in the `CLOUD_PARAMS` is the subscription_id of the Azure function app.

   This creates the `AZURE_DEMO_CATALOG` catalog and catalogs all the functions of the provided `SUBSCRIPTION_ID`.

   See CREATE_CATALOG Procedure for more information. You can query DBA_CLOUD_FUNCTION_CATALOG View and USER_CLOUD_FUNCTION_CATALOG View views to retrieve the list of all the catalogs in your database.

3. You can list the functions in a catalog.

   Following is the example to list Azure functions:

```
VAR function_list CLOB;

BEGIN
 DBMS_CLOUD_FUNCTION.LIST_FUNCTIONS (
    credential_name  => 'AZURE$PA',
    catalog_name     => 'AZURE_DEMO_CATALOG',
```

```
    function_list    => :function_list
);
END;
/

SELECT JSON_QUERY (:function_list, '$' RETURNING VARCHAR2(32676) pretty)
AS search_results FROM dual;
```

See LIST_FUNCTIONS Procedure for more information.

4. Run the `DBMS_CLOUD_FUNCTION.SYNC_FUNCTIONS` to create wrapper SQL functions. You can use one of the following methods to create the wrapper SQL functions in the catalog, that call their respective cloud functions:

   - **SYNC_FUNCTIONS**: `SYNC_FUNCTIONS` is the quickest and simplest method, which automatically syncs (creates or deletes) wrapper functions in the catalog with the complete list of Azure functions. For example:

```
BEGIN
 DBMS_CLOUD_FUNCTION.SYNC_FUNCTIONS (
        catalog_name => 'AZURE_DEMO_CATALOG'
 );
END;
/
```

This creates a PL/SQL wrapper for adding new functions to the catalog and removing wrappers for functions that have been deleted from the catalog.

Run the following query to verify the sync.

```
SELECT object_name FROM sys.all_objects WHERE owner='TEST_USER' AND
object_type='FUNCTION';
```

> **Note:**
>
> Keep a note of the current user in order to run this command.

See SYNC_FUNCTIONS Procedure for more information.

- You can manually create a `SQL` Function in your catalog that calls its respective Azure function using `AZURE_DEMO_CATALOG`.

Example to create a function in the `AZURE_DEMO_CATALOG` catalog.

```
BEGIN
DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
  credential_name => 'AZURE$PA',
  catalog_name    => 'AZURE_DEMO_CATALOG',
  function_name   => 'azure_testfunc',
  function_id     => 'function_id_path',
  input_args      => :function_args
);
```

```
END;
/
```

> **Note:**
>
> The maximum length of the function name is limited to 100 characters.

This creates the `AZURE_TESTFUNC` function in the `AZURE_DEMO_CATALOG` catalog.

The `AZURE_TESTFUNC` function in the catalog is a reference to the respective Azure function whose subscription is referenced by the `FUNCTION_ID` parameter. Invoking the function in the catalog along with its arguments runs the corresponding Azure function and provides the output returned by the function.

You can query DBA_CLOUD_FUNCTION View and USER_CLOUD_FUNCTION View views to retrieve the list of all the functions in your database.

See CREATE_FUNCTION Procedure for more information.

See Azure Functions HTTP trigger for more information.

5. After the function is created you can `DESCRIBE` and invoke it.

```
DESC AZURE_TESTFUNC


SELECT AZURE_TESTFUNC(NULL) FROM dual;
```

This invokes the `AZURE_TESTFUNC` function by calling the function reference `/subscriptions/44496e556-8ssp-4262-b389-0f15f685c879/resources/ADBStest/providers/Microsoft.Web/sites/AZUREADBS/functions/HttpTrigger_example` in the `AZURE_DEMO_CATALOG` catalog.

6. You can drop an existing function using `DROP_FUNCTION` procedure. For example:

```
EXEC DBMS_CLOUD_FUNCTION.DROP_FUNCTION (CATALOG_NAME =>
'AZURE_DEMO_CATALOG', FUNCTION_NAME => 'AZURE_TESTFUNC');
```

This drops the `AZURE_TESTFUNC` function from the `AZURE_DEMO_CATALOG` catalog.

See DROP_FUNCTION Procedure for more information.

7. You can drop an existing catalog using `DROP_CATALOG` procedure. For example:

```
BEGIN
    DBMS_CLOUD_FUNCTION.DROP_CATALOG (
      catalog_name     => 'AZURE_DEMO_CATALOG'
  );
END;
/
```

This drops the `AZURE_DEMO_CATALOG` from your database.

See DROP_CATALOG Procedure for more information.

Invoke External Procedures as SQL Functions

Shows the steps to invoke external procedures using PL/SQL within your database.

- External Procedures Overview
  External procedures are functions written in a third-generation language and callable from within PL/SQL or SQL as if they were a PL/SQL routine or function.
- About Using External Procedures in Autonomous Database
  You can invoke and use external procedures in your Autonomous Database with user defined functions.
- Define the C Procedure
  Define the C procedure using one of these prototypes.
- Create a Shared Library (.so) File
  Create a shared object (`.so` file) library. The shared object library contains the C procedure (external procedure) which was defined in the previous step.
- Get the OCI Marketplace EXTPROC Stack Application
  Shows the steps to get the OCI Marketplace `EXTPROC` Stack Application.
- Launch EXTPROC Stack Application
  Launch the `EXTPROC` Stack Application from the EXTPROC Application Details page.
- Create Stack for EXTPROC Agent Application
  Shows the steps to create Stack for `EXTPROC` instance.
- Upload Wallet to Create Secure Connection to the EXTPROC Agent Instance
  A self-signed wallet is created as part of the `EXTPROC` agent application creation. This wallet allows you to access the `Extrpoc` agent instance.
- Steps to Invoke an External Procedure as a SQL Function
  Shows the steps to invoke an external Procedure as a SQL function.

External Procedures Overview

External procedures are functions written in a third-generation language and callable from within PL/SQL or SQL as if they were a PL/SQL routine or function.

External procedures promote reusability, efficiency, and modularity. Existing dynamic link libraries (DLLs) written in other languages can be called from PL/SQL programs. The DLLs are loaded only when needed and they can be enhanced without affecting the calling programs.

Using external procedures also enhances performance, because third-generation languages perform certain tasks more efficiently than PL/SQL, which is better suited for SQL transaction processing.

External procedures are useful when:

- Solving scientific and engineering problems
- Analyzing data
- Controlling real-time devices and processes

See What Is an External Procedure? for more information.

About Using External Procedures in Autonomous Database

You can invoke and use external procedures in your Autonomous Database with user defined functions.

You do not install external procedures on an Autonomous Database instance. To use an external procedure, the procedure is hosted remotely on a VM running in an Oracle Cloud Infrastructure Virtual Cloud Network (VCN).

External procedures are only supported when your Autonomous Database is on a private endpoint. The `EXTPROC` agent instance is hosted on a private subnet and the Autonomous Database access the `EXTPROC` agent through a Reverse Connection Endpoint (RCE).

> **Note:**
>
> Autonomous Database only supports C language external procedures.

External procedures are deployed by using:

- An Oracle provided container image with `EXTPROC` agent installed and configured as a part of the Oracle Cloud Infrastructure (OCI) Marketplace stack.

  The `EXTPROC` agent instance is hosted remotely on a VM running in an Oracle Cloud Infrastructure Virtual Cloud Network (VCN). The secure communication between your Autonomous Database and the `EXTPROC` agent instance is ensured by setting Network Security Group (NSG) rules such that the traffic is allowed from your Autonomous Database instance running on a private endpoint to the `EXTPROC` agent instance.

  The `EXTPROC` agent image is pre-configured to host and execute external procedures on port 16000.
- PL/SQL procedures to create a library and to register and invoke external functions and procedures.

  See DBMS_CLOUD_FUNCTION Package for more information.

Follow these steps to invoke an external procedure on Autonomous Database:

- Define the C procedure. See Define the C Procedure.
- Create a shared object (`.so` file) library. See Create a Shared Library (.so) File.
- Launch the Autonomous Database `EXTPROC` stack application. See Get the OCI Marketplace EXTPROC Stack Application.
- Provision and configure Oracle Autonomous Database `EXTPROC` agent. See Create Stack for EXTPROC Agent Application for more information.
- Configure your Autonomous Database to connect to the `EXTPROC` agent instance. See Upload Wallet to Create Secure Connection to the EXTPROC Agent Instance for more information.
- Create a remote library using `DBMS_CLOUD_FUNCTION.CREATE_CATALOG`. See Steps to Invoke an External Procedure as a SQL Function for more information.
- Use the user defined function you created in the previous step. See Steps to Invoke an External Procedure as a SQL Function for more information.

Define the C Procedure

Define the C procedure using one of these prototypes.

- Kernighan & Ritchie style prototypes. For example:

```
void UpdateSalary(x)
 float x;
...
```

- ISO/ANSI prototypes other than numeric data types that are less than full width (such as `float`, `short`, `char`). For example:

```
void UpdateSalary(double x)
...
```

- Other data types that do not change size under default argument promotions.

  This example changes size under default argument promotions:

```
void UpdateSalary(float x)
...
```

Create a Shared Library (.so) File

Create a shared object (`.so` file) library. The shared object library contains the C procedure (external procedure) which was defined in the previous step.

You generate a shared object library using the following command:

```
gcc -I/u01/app/oracle/extproc_libs/ -shared -fPIC -o extproc.so UpdateSalary.c
```

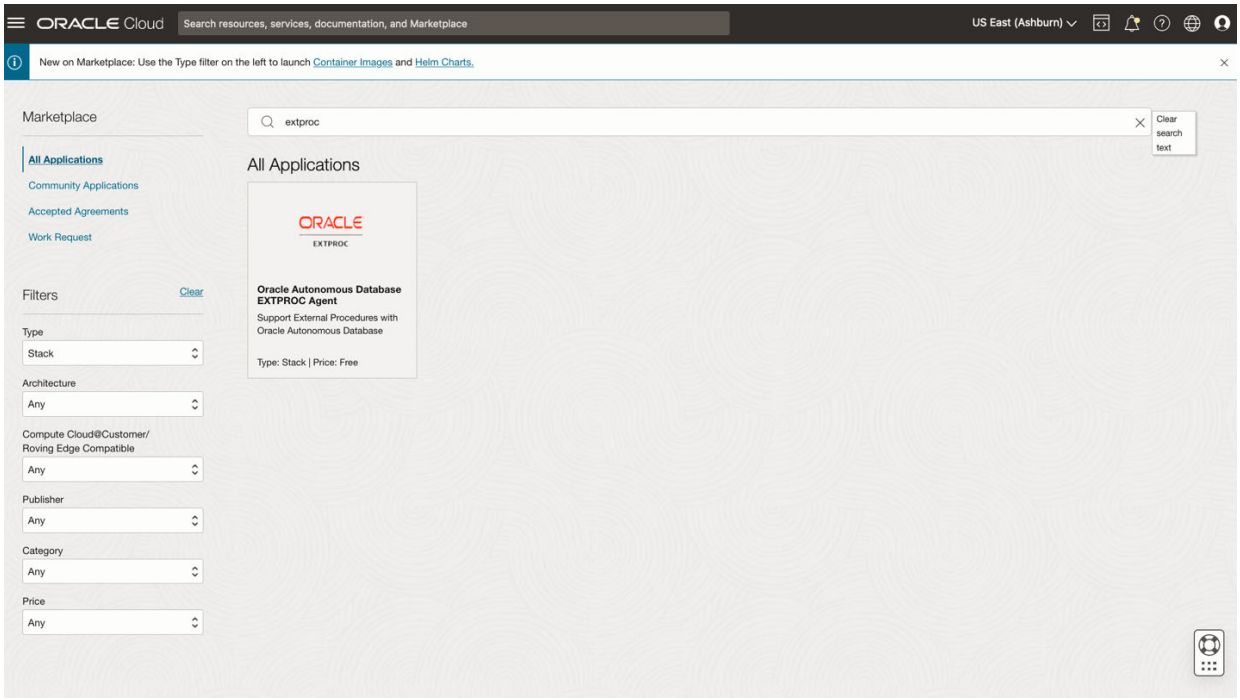
This creates the shared object (`.so`), `extproc.so` library. The `UpdateSalary` procedure, defined in the previous step, is contained in the `extproc.so` library. The shared object (`.so`) libraries are dynamically loaded at run time.

Get the OCI Marketplace EXTPROC Stack Application

Shows the steps to get the OCI Marketplace `EXTPROC` Stack Application.

Perform the following steps:

1. Sign in to the OCI Console at http://cloud.oracle.com. See Sign in to the Oracle Cloud Infrastructure Console for more information.
2. From the Oracle Cloud Infrastructure left navigation menu click **Marketplace** and then, under **Marketplace** click **All Applications**. This takes you to the Marketplace All Applications dashboard.
3. Enter `EXTPROC` in the search field and click **search**.
4. Click the `EXTPROC` widget of **Type: Stack**.



This takes you to the **Oracle Autonomous Database EXTPROC Agent** details page.

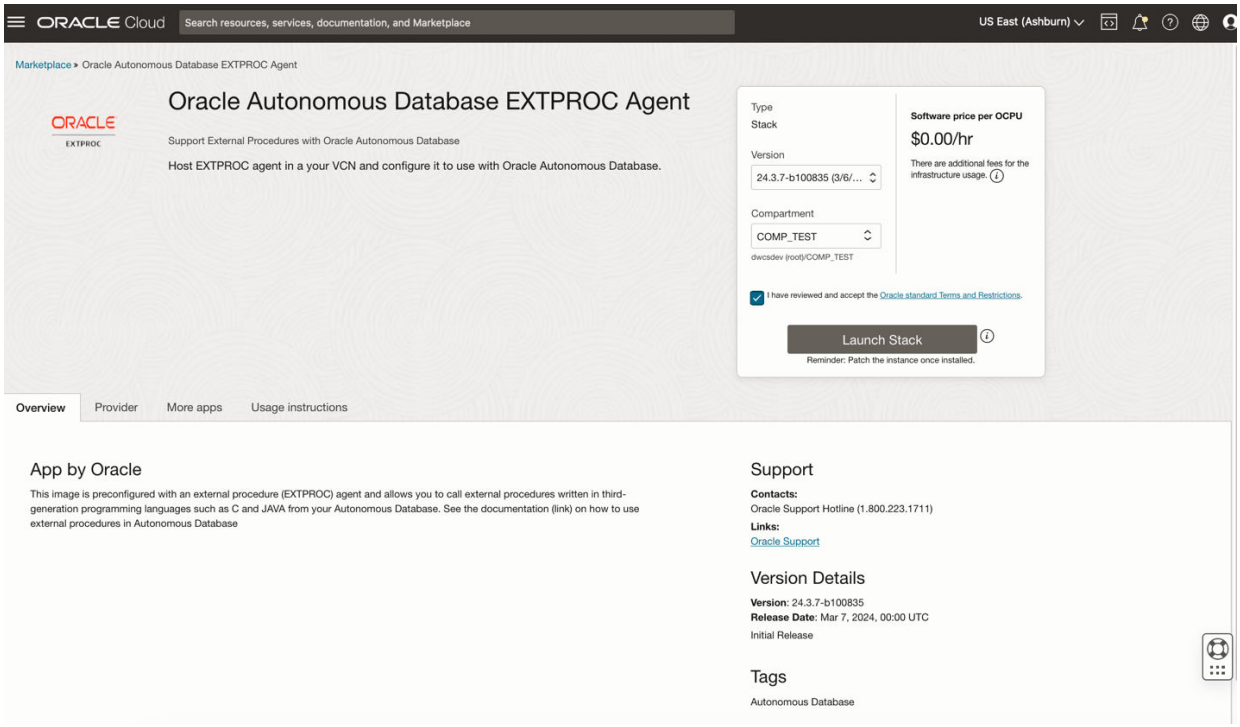
Launch EXTPROC Stack Application

Launch the `EXTPROC` Stack Application from the EXTPROC Application Details page.

1. On the Oracle Autonomous Database `EXTPROC` Agent page, under **Type Stack**, perform the following:
   - From the **Version** drop-down list, select the package version of the stack. By default, the menu displays the latest version.
   - From the **Compartment** drop-down list, select the name of the compartment where you want to launch the instance.

     > **Note:**
     >
     > If you don't have permission to launch the instance in the selected compartment, the instance is launched in the root compartment.

   - Select the **I have reviewed and accept the Oracle Standard Terms and Restrictions** checkbox.
2. Click **Launch Stack**.



This takes you to the **Create stack** page that allows you to create stack for the `EXTPROC` agent.

Create Stack for EXTPROC Agent Application

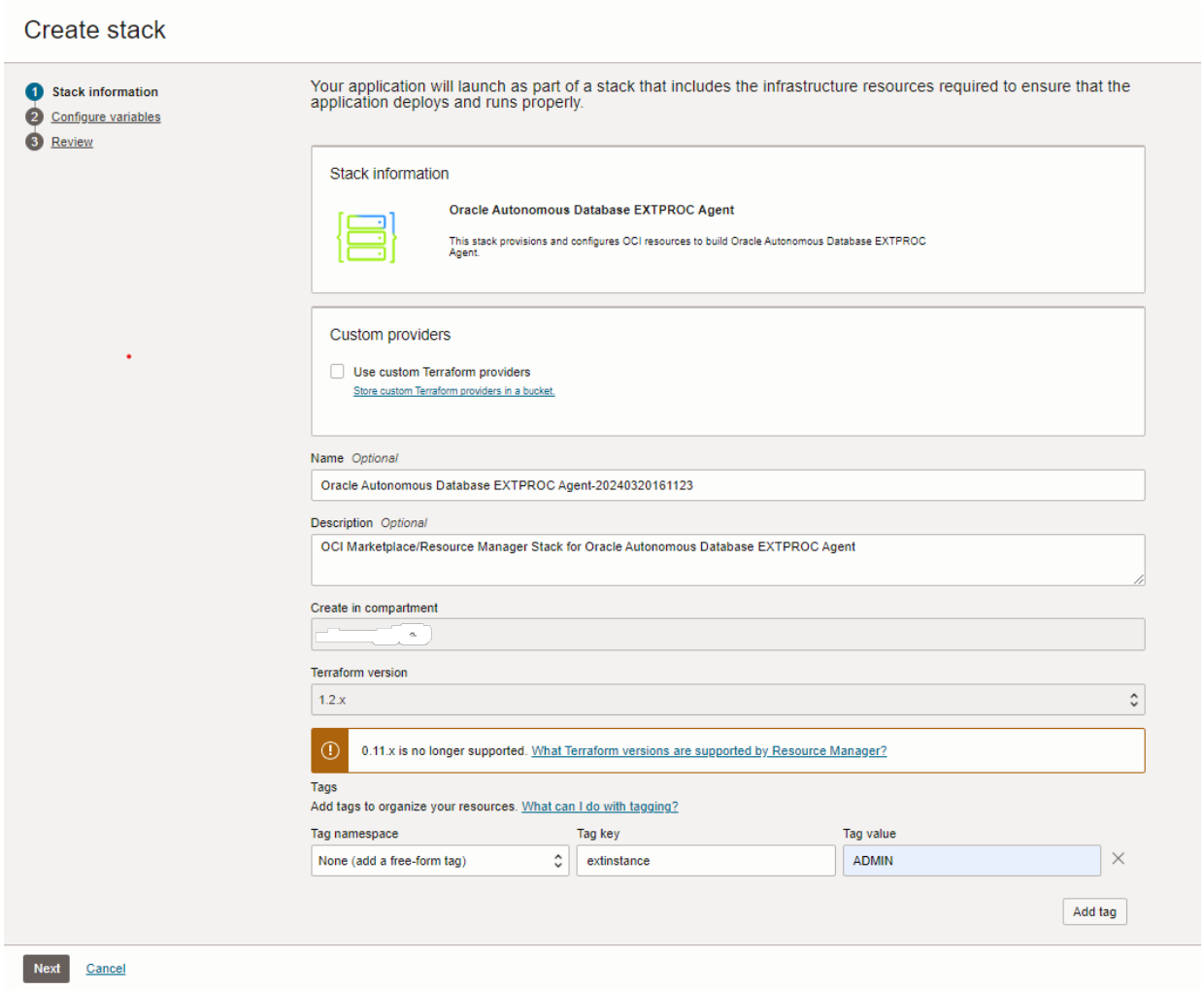
Shows the steps to create Stack for `EXTPROC` instance.

In the **Create stack** wizard, perform the following steps:

1. On the **Stack information** page, review and edit the following information as necessary:
   - **Stack information**
   - **Custom providers**
   - **Name (Optional)**: You can edit the default stack name. Avoid entering confidential information.
   - **Description (Optional)**: You can edit the default stack description. Avoid entering confidential information.
   - **Create in compartment**
   - **Terraform version**
   - **Tags**: Provide the following to assign tags to the stack.
     - **Tag namespace**: To add a defined tag, select an existing namespace. To add a free-from tag, leave the value blank.
     - **Tag key**: To add a defined tag, select an existing tag key. To add a free-form tag, type the key name that you want.
     - **Tag value**: Type the tag value that you want.

     Add tag: Click to add another tag.

     See Resource Tags for more information on tagging.



2. Click **Next**.

   This takes you to the **Configure variables** page which enables you to configure variables for the infrastructure resources that the stack creates when you run the apply job for this execution plan.

3. On the Configure variables page enter the information in the areas: **Configure the EXTPROC Agent**, **Network Configuration**, and **Compute configuration**.

   a. Provide information in the **Configure the EXTPROC Agent** area.

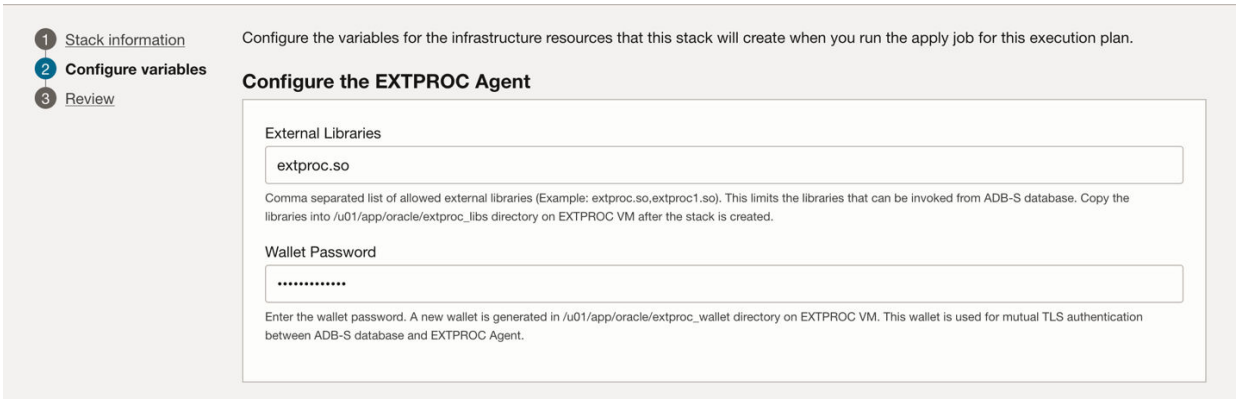
      - **External Libraries**: Provide a list of libraries, separated by comma (,), which you want to allow to be invoked from your Autonomous Database. For example, `extproc.so`, `extproc1.so`.

        After you create the stack, you must copy the libraries to the `/u01/app/oracle/extproc_libs` directory on the `EXTPROC` agent VM.

      - **Wallet Password:** Provide the wallet password.

        The wallet and a self-signed certificate is generated for mutual TLS authentication between the Autonomous Database and the `EXTPROC` agent VM. The wallet is created in the `/u01/app/oracle/extproc_wallet` directory.

> **Note:**
>
> After the wallet is created, the wallet password cannot be changed.

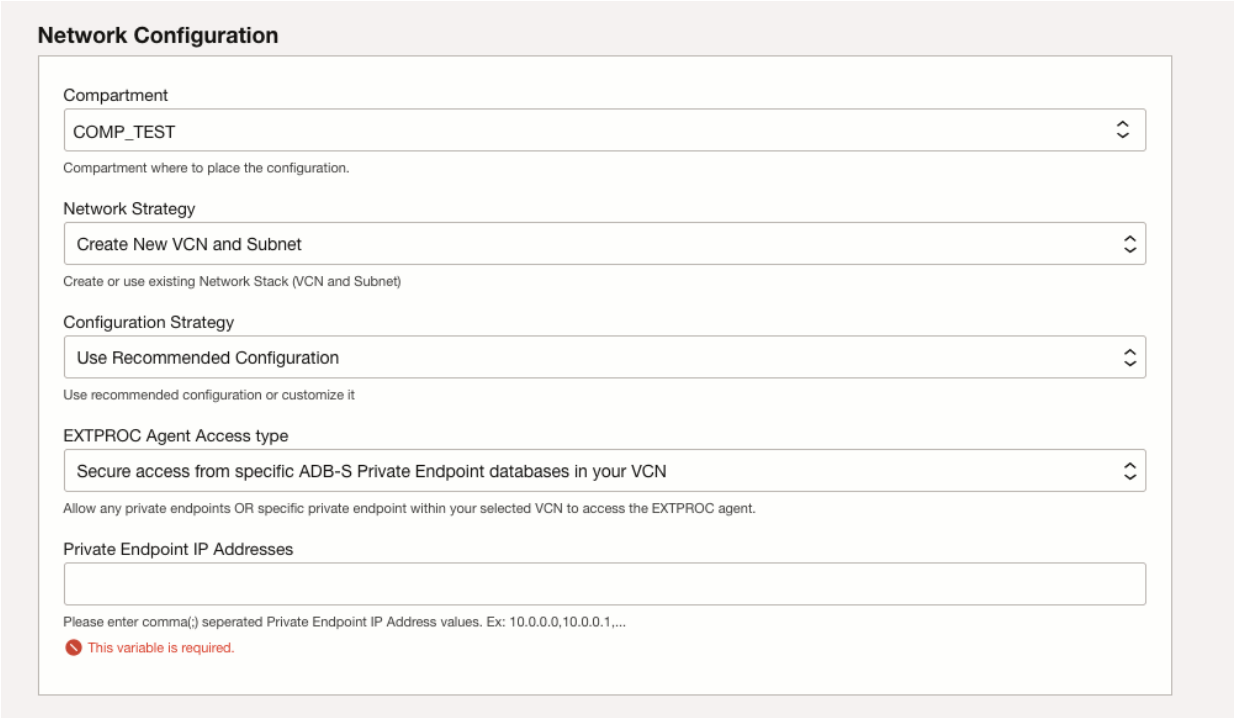


b. Provide information in the **Network Configuration** area.

- **Compartment**: From the drop-down list, choose the compartment where you want to place the configuration.
- **Network Strategy**: Choose one of the options from the drop-down list, **Create New VCN and Subnet** or **Use Existing VCN and Subnet**.
  - **Create New VCN and Subnet**: Choose this option if a private endpoint is not configured for your Autonomous Database. This creates a new VCN with public and private subnet that are preconfigured with security rules.

    If you select this option the page also shows the Configuration Strategy drop-down list:

    Choose **Use Recommended Configuration** from the **Configuration Strategy** drop-down list.



- **Use Existing VCN and Subnet**: Select this option to create the EXTPROC agent using an existing VCN. This creates the EXTPROC agent instance in the provided subnet.

  When you select this option, provide the following information for the existing VCN and Subnet:

  * Under **Virtual Cloud Network**:

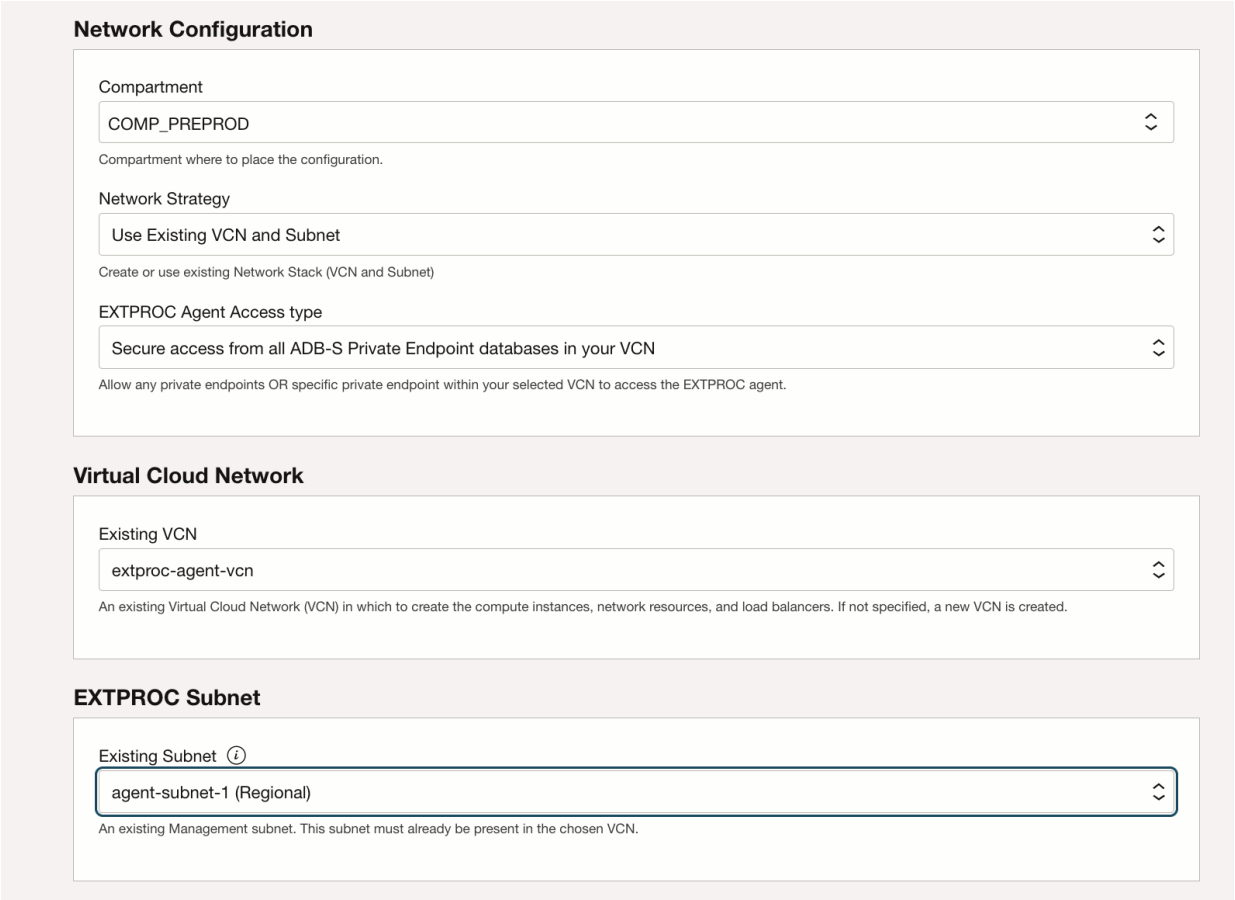
    From the **Existing VCN** drop-down list choose an existing VCN. If the specified VCN does not exist, a new VCN is created.

  * Under **EXTPROC Subnet**:

    From the **Existing Subnet** drop-down list choose an existing subnet.

    When you choose to use an existing VCN and subnet, add an ingress rule for the `EXTPROC` agent instance's port `16000`. You also add an egress rule on public subnet.

    See Configuring Network Access with Private Endpoints for more information.



- **EXTPROC Agent Access type**: Choose one of the following options from the drop-down list.
  - **Secure access from specific ADB-S Private Endpoint databases in your VCN**: Choose this option to allow only specified private endpoint IPs inside your Virtual Cloud Network (VCN) to connect to your `EXTPROC` agent.

    When this option is chosen, you provide a list of allowed private endpoint IP Addresses in the next step.
  - **Secure access from all ADB-S Private Endpoint databases in your VCN**: Choose this option to allow any private endpoint inside your Virtual Cloud Network (VCN) to connect to your `EXTPROC` agent.
- **Private Endpoint IP Addresses**

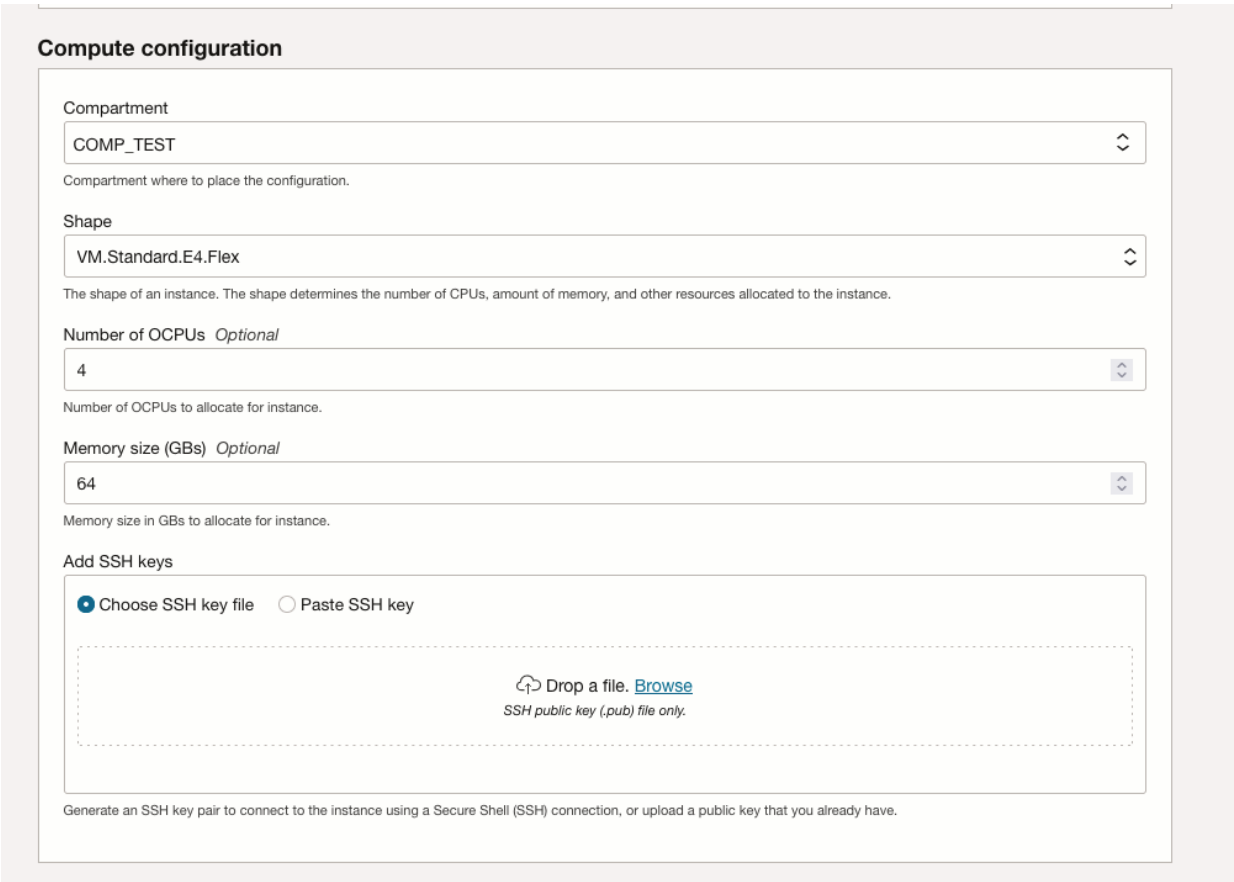
  Provide a list of private endpoint IP addresses separated by comma (,) for the Private Endpoint IP Addresses variable. For example, `10.0.0.0`, `10.0.0.1`.

> **Note:**
>
> This field only shows when you select **Secure access from specific ADB-S Private Endpoint databases in your VCN** for the **EXTPROC Agent Access type**.

c. Provide the **Compute configuration** information.

- **Compartment**: Select the compartment where you want to create the stack.

- **Shape**: Select a shape based on the workload requirements of the `EXTPROC` agent instance. The shape determines the resources allocated to the `EXTPROC` agent instance.
- **Number of OCPUs**: Choose the number of OCPUs that you want to allocate to the `EXTPROC` agent instance.
- **Memory size (GBs)**: Choose the amount of memory in Gigabytes (GB) that you want to allocate to the `EXTPROC` agent instance.
- **Add SSH keys**: Upload an SSH public key or paste the public key. Select one of the following options:
  - **Choose SSH key file**: Upload the public key portion of your key pair. Either browse to the key file that you want to upload, or drag and drop the file into the box.
  - **Paste SSH key**: Paste the public key portion of your key pair in the box.

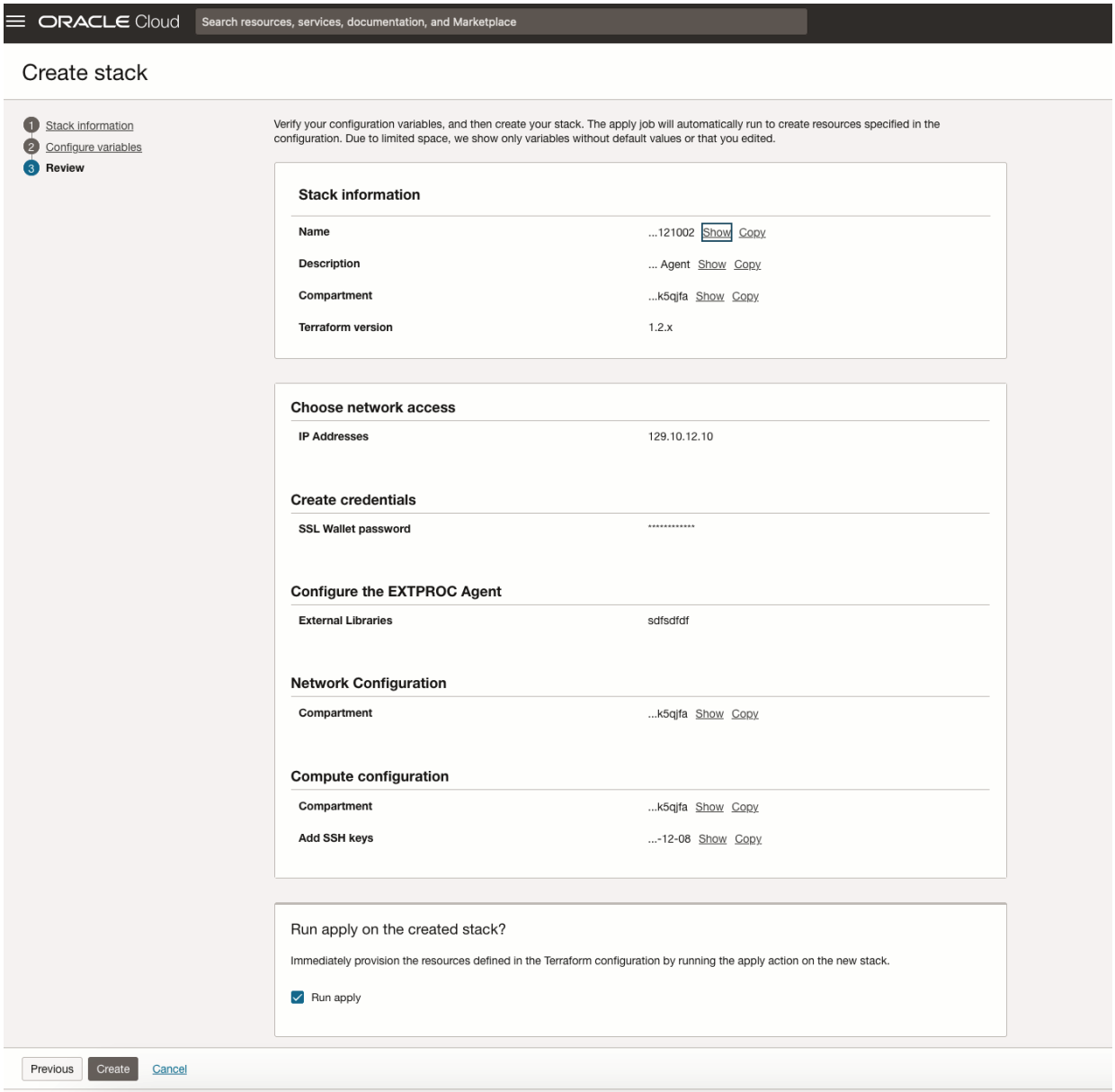


4. Click **Next**.

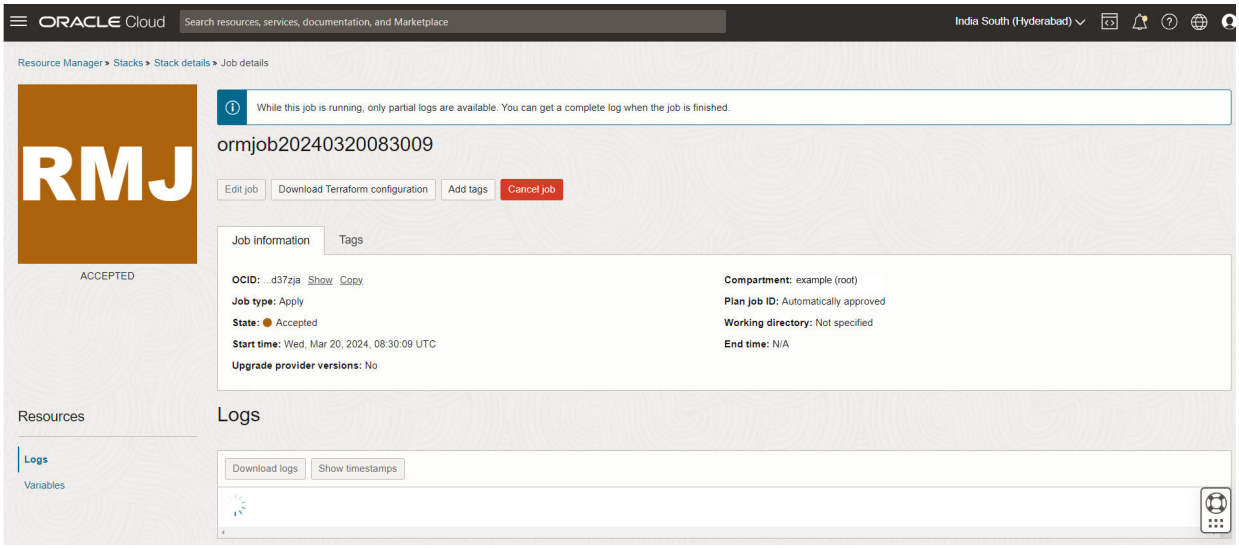
   This takes you to the **Review** page.

5. On the **Review** page perform the following steps:
   a. Verify the configuration variables.
   b. Select the **Run apply** checkbox under **Run apply on the created stack?**
   c. Click **Create**.

> **Note:**
>
> This area does not show variables that have default values or variables that you have not changed.



Resource Manager runs the apply job to create stack resources accordingly. This takes you to the **Job details** page and the job state is **Accepted**. When the apply job starts the status is updated to **In Progress**.



> **Note:**
>
> The information you need to connect to the instance created as part of the stack is available in the **Application Information** tab.

Upload Wallet to Create Secure Connection to the EXTPROC Agent Instance

A self-signed wallet is created as part of the `EXTPROC` agent application creation. This wallet allows you to access the `Extrpoc` agent instance.

To execute remote procedures at the `EXTPROC` agent instance, the Autonomous Database and the `EXTPROC` agent connect using Mutual Transport Layer Security (mTLS). When using Mutual Transport Layer Security (mTLS), clients connect through a TCPS (Secure TCP) database connection using standard TLS 1.2 with a trusted client certificate authority (CA) certificate. See About Mutual TLS (mTLS) Authentication for more information.

> **Note:**
>
> You can also obtain and use a public certificate issued by a Certificate Authority (CA).

As a prerequisite, you must export the wallet to Object Storage from the `/u01/app/oracle/extproc_wallet` directory on the VM where `EXTPROC` runs.

Follow these steps to upload the wallet to your Autonomous Database:

1. Import the wallet, `cwallet.sso`, containing the certificates for the `EXTPROC` agent instance from Object Storage in your Autonomous Database. Note the following for the wallet file:
   - The wallet file, along with the Database user ID and password provide access to the `EXTPROC` agent instance. Store wallet files in a secure location and share them only with authorized users.
   - Do not rename the wallet file. The wallet file in Object Storage must be named `cwallet.sso`.

2. Create credentials to access your Object Storage where you store the wallet file `cwallet.sso`. See CREATE_CREDENTIAL Procedure for information about the username and password parameters for different object storage services.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See About Using Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Create a directory on Autonomous Database for the wallet file `cwallet.sso`.

   ```
   CREATE DIRECTORY wallet_dir AS 'directory_location';
   ```

   See Create Directory in Autonomous Database for more information creating directories.

4. Use `DBMS_CLOUD.GET_OBJECT` to upload the wallet. For example:

   ```
   BEGIN
     DBMS_CLOUD.GET_OBJECT (
       credential_name     => 'DEF_CRED_NAME',
       object_uri          => 'https://objectstorage.us-
   phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/cwallet.sso',
       directory_name      => 'WALLET_DIR'
   );
   END;
   /
   ```

   In this example, `namespace-string` is the Oracle Cloud Infrastructure Object Storage namespace and `bucketname` is the bucket name. See Object Storage Namespaces for more information.

   The wallet is copied to the directory created in the previous step, *WALLET_DIR*. The wallet that allows you to connect to the `EXTPROC` agent instance is now available on your Autonomous Database instance.

Steps to Invoke an External Procedure as a SQL Function

Shows the steps to invoke an external Procedure as a SQL function.

After you launch the OCI Marketplace `EXTPROC` stack application and configure it to run external procedures, you create a library of SQL wrapper functions that reference and call their respective external procedures.

As a prerequisite, the whitelisted libraries must be copied into the `/u01/app/oracle/extproc_libs` directory on the `EXTPROC` VM.

Follow these steps to create a library in your Autonomous Database and register C routines as an external procedure in the library:

1. Create a library.

   An external procedure is a C language routine stored in a library. To invoke external procedures with Autonomous Database, you create a library.

   Run `DBMS_CLOUD_FUNCTION.CREATE_CATALOG` to create a library. For example:

   ```
   BEGIN
       DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
           library_name               => 'demolib',
   ```

```
        library_listener_url       => 'remote_extproc_hostname:16000',
        library_wallet_dir_name    => 'wallet_dir',
        library_ssl_server_cert_dn => 'CN=VM Hostname',
        library_remote_path        => '/u01/app/oracle/extproc_libs/
library name'
);
END;
/
```

This creates the `demolib` library in your Autonomous Database and registers the dynamic link library in your database. The `EXTPROC` agent instance is pre-configured to host external procedures on port 16000.

See CREATE_CATALOG Procedure for more information.

Query DBA_CLOUD_FUNCTION_CATALOG View and USER_CLOUD_FUNCTION_CATALOG View views to retrieve the list of all the catalogs and libraries in your database.

Query the USER_CLOUD_FUNCTION_ERRORS View view to list any errors generated during the connection validation to the remote library location.

2. After you create the library, use `DBMS_CLOUD_FUNCTION.CREATE_FUNCTION` to create PL/SQL wrapper functions that refer to the external procedures (C functions).

   Example:

```
DECLARE
    plsql_params clob    := TO_CLOB('{"sal": "IN, FLOAT", "comm" :"IN,
FLOAT"}');
    external_params clob := TO_CLOB('sal FLOAT, sal INDICATOR SHORT, comm
FLOAT, comm INDICATOR SHORT,
    RETURN INDICATOR SHORT, RETURN FLOAT');
BEGIN
DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
    LIBRARY_NAME    => 'demolib',
    FUNCTION_NAME   => '"PercentComm"',
    PLSQL_PARAMS    => plsql_params,
    EXTERNAL_PARAMS => external_params,
    API_TYPE        => 'FUNCTION',
    WITH_CONTEXT    => FALSE,
    RETURN_TYPE     => 'FLOAT'
);
END;
/
```

   This creates the `PercentComm` function and registers the `PercentComm` external procedure in the `DEMOLIB` library.

   The `PercentComm` function in the library is a reference to the respective external procedure whose name is referenced by the `FUNCTION_ID` parameter.

   In this example the `FUNCTION_ID` parameter is not provided, the value provided for the `FUNCTION_NAME` parameter is used as a `FUNCTION_ID`.

Example:

```
DECLARE
    plsql_params clob := TO_CLOB('{"row_id": "IN,CHAR"}');
    external_params clob := TO_CLOB('CONTEXT, row_id STRING, row_id LENGTH
SB4');
BEGIN
DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
    LIBRARY_NAME    => 'demolib',
    FUNCTION_NAME   => 'UpdateSalary_local',
    FUNCTION_ID     => '"UpdateSalary"',
    PLSQL_PARAMS    => plsql_params,
    EXTERNAL_PARAMS => external_params,
    API_TYPE        => 'PROCEDURE',
    WITH_CONTEXT    => TRUE
);
END;
/
```

This creates the `UPDATESALARY_LOCAL` procedure and registers the `UpdateSalary` procedure in the `DEMOLIB` library.

The `UPDATESALARY_LOCAL` procedure in the library is a reference to the respective external procedure `UpdateSalary` whose name is referenced by the `FUNCTION_ID` parameter.

You can query DBA_CLOUD_FUNCTION View and USER_CLOUD_FUNCTION View views to retrieve the list of all the functions that are available to use in your database.

See CREATE_FUNCTION Procedure for more information.

3. After creating the function, you can `DESCRIBE` it. For example:

```
DESC "PercentComm";
```

4. You can drop an existing function using `DROP_FUNCTION` procedure. For example:

```
EXEC DBMS_CLOUD_FUNCTION.DROP_FUNCTION (LIBRARY_NAME => 'demolib',
FUNCTION_NAME => '"PercentComm"');
```

This drops the `PercentComm` function from the `DEMOLIB` library.

See DROP_FUNCTION Procedure for more information.

5. You can drop an existing library using `DROP_CATALOG` procedure. For example:

```
BEGIN
    DBMS_CLOUD_FUNCTION.DROP_CATALOG (
      LIBRARY_NAME  => 'demolib'
  );
END;
/
```

This drops the `DEMOLIB` library.

See DROP_CATALOG Procedure for more information.

Use Cloud Tables to Store Logging and Diagnostic Information

You can create Cloud Tables where table data resides on Oracle managed Cloud Storage and the table data does not consume database storage.

- About Cloud Tables
  You can create Cloud Tables as a complementary alternative to in-database tables. All Cloud Table data is stored in Oracle managed Object Storage. Oracle managed Object Storage is external storage, outside of the database, that Autonomous Database creates and manages.
- Create Cloud Tables
  Shows the steps to create a Cloud Table on Autonomous Database.
- Cloud Table Notes
  Provides notes for Cloud Tables:

About Cloud Tables

You can create Cloud Tables as a complementary alternative to in-database tables. All Cloud Table data is stored in Oracle managed Object Storage. Oracle managed Object Storage is external storage, outside of the database, that Autonomous Database creates and manages.

You can use Cloud Tables to store infrequently used application logging data, diagnostic information, or to store other data. In some existing applications that do not run on Autonomous Database you might store this kind of information in files on a local file system (for example using `UTL_FILE` APIs). Such logging mechanisms and the associated files can be very helpful when you need to diagnose and resolve application errors. However, storing information in database tables can use large amounts of database storage for data that is infrequently used. Using Cloud Tables the persistent data is saved in Oracle managed Object Storage, without consuming database storage.

SELECT and DML Restrictions for Cloud Tables

Cloud Tables function like ordinary database tables with some restrictions. You can use SELECT and DML, data manipulation statements, with the following exceptions:

- `MERGE` statements are not supported.
- `LOB` columns are limited to 10MB of data.
- DML concurrency control is different, and therefore:
  - `LOCK TABLE` may not prevent concurrent DML as it does for a database table.
  - `INSERT` does not acquire a lock on the table, and therefore INSERT is never blocked by concurrent DML operations.
  - `UPDATE` and `DELETE` operations both acquire an exclusive lock on a Cloud Table. Therefore, `UPDATE` or `DELETE` transactions block concurrent `UPDATE` or `DELETE` operations on a Cloud Table.
- Only NOT NULL constraints are enforced.
- DML is allowed in a Read-Write Autonomous Database as it is for any other table; Cloud Tables also allow DML operations in a Refreshable Clone.

Cloud Tables do not support the following:

- Indexes

- Invisible columns
- Virtual columns
- DML triggers
- More than 996 columns
- Boolean data type columns

Lifecycle Management Operations and Cloud Tables

Cloud Table data is stored in Oracle managed Object Storage. This means certain operations on Autonomous Database handle Cloud Tables differently than in-database tables, as follows:

- Cloud Table data is excluded from an Autonomous Database instance's backup and recovery (the data is not backed up and you cannot restore Cloud Table data).
- Cloud Table Data is protected through Oracle managed Object Storage.
- The lifecycle management operations that impact the state of an Autonomous Database instance do not have an impact on the data stored in Cloud Tables.

Cloud Table naming in Object Storage is defined uniquely for each Autonomous Database instance, based on its OCID. This means that any operation that changes or introduces a new OCID for an existing database has an impact on Cloud Tables. The following illustrates the impact of lifecycle operations on Cloud Table data.

Lifecycle Operation	Cloud Table Data Availability
Same region database clone	Cloud Table is cloned without Cloud Table data
Cross-region database clone	Cloud Table is cloned without Cloud Table data
Same region (local) Autonomous Data Guard Standby	Cloud Table and Cloud Table data are accessible
Cross-region Autonomous Data Guard Standby	Cloud Table is available on the standby, without the Cloud Table data
Same region (local) Backup-Based Disaster Recovery peer	Cloud Table and Cloud Table data are accessible
Cross-region Backup-Based Disaster Recovery peer	Cloud Table is available on the standby, without Cloud Table data
Lifecycle management operations impacting the SCN/timestamp of an Autonomous Database instance, including: • Long term backup • Restore database (point in time restore) • Clone from backup	Cloud Table will continue to be updated and the old state of Cloud Table data is not preserved or restored. This means only the current Cloud Table data is available.
Lifecycle Management operations, including: • Manage resource allocation • Move • Shrink • Rename • Mode: Read-only/read-write • Change workload type: for example from Data Warehouse to Transaction Processing	No impact on Cloud Tables or on Cloud Table data

Create Cloud Tables

Shows the steps to create a Cloud Table on Autonomous Database.

To create a Cloud Table:

1. Run the `CREATE_CLOUD_TABLE` procedure.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CLOUD_TABLE(
      table_name  => 'CLOUD_TABLE_TEST',
      column_list => 'I INTEGER, STR1 VARCHAR2(32)' );
   END;
   /
   ```

   See CREATE_CLOUD_TABLE Procedure for more information.

2. Insert data into the Cloud Table.

   ```
   INSERT INTO cloud_table_test VALUES (1, 'xyz');
   ```

3. Select data from a Cloud Table.

   ```
   SELECT * FROM cloud_table_test;


   I          STR1
   ---------- --------------------------------
   1          xyz
   ```

Use `DROP TABLE` when you want to drop a Cloud Table.

For example:

```
DROP TABLE CLOUD_TABLE_TEST;
```

Cloud Tables do not support the recycle bin.

See Cloud Table Notes for additional information.

Cloud Table Notes

Provides notes for Cloud Tables:

- You can grant `SELECT`, `INSERT`, and `UPDATE` privileges for a Cloud Table. No other privileges can be granted to a Cloud Table.

  See Configuring Privilege and Role Authorization for more information.

- Cloud Table constraints are limited to constraints in `RELY DISABLE NOVALIDATE` mode, which means the constraint is not enforced. The only exception to this is for `NOT NULL` constraints.

  Cloud Tables support all `NOT NULL` constraint modes including the default mode (`ENABLE VALIDATE`). `PRIMARY KEY`, `UNIQUE`, `FOREIGN KEY`, and `NOT NULL` constraints are supported; `CHECK` constraints are not supported.

  You can declare constraints inline as part of `COLUMN_LIST`.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CLOUD_TABLE(
        table_name  => 'CLOUD_TAB_WITH_CONSTRAINTS',
        column_list => 'PK INTEGER,
            DATE_ID INT REFERENCES DATE_DIM(DATE_ID) RELY DISABLE
NOVALIDATE,
            VAL NUMBER NOT NULL,
            CONSTRAINT CLOUD_TAB_PK PRIMARY KEY(PK) RELY DISABLE
NOVALIDATE');
END;
/
```

See Cloud Table Notes for additional Cloud Table limitations.

- The `DBMS_CLOUD` package is an invoker's rights package. The `DBMS_CLOUD.CREATE_CLOUD_TABLE` procedure only allows you to create a table in the invoker's schema.

  See Invoker’s Rights and Definer’s Rights Clausefor more information.

- The `column_list` parameter in a `DBMS_CLOUD.CREATE_CLOUD_TABLE` procedure call can include the `DEFAULT` clause, which functions like the `DEFAULT` clause in `CREATE TABLE`. See CREATE TABLE for more information.

  For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CLOUD_TABLE(
   table_name  => 'CLOUD_TABLE_TEST_DEFAULT',
   column_list => 'I INTEGER, STR2 VARCHAR2(32) DEFAULT ''ABC''');
END;
/
```

  Then you can insert with default values. For example:

```
INSERT INTO cloud_table_test_default (i) VALUES (1);
1 row created.
INSERT INTO cloud_table_test_default VALUES (2, default);
1 row created.
INSERT INTO cloud_table_test_default VALUES (3, null);
1 row created.
INSERT INTO cloud_table_test_default VALUES (4, 'xyz');
1 row created.
COMMIT;
Commit complete.
SELECT * FROM cloud_table_test_default ORDER BY i;

I STR2
- ----
1 ABC
2 ABC
3 null
4 xyz
```

Analyze

Provides information on the built-in tools that allow you to explore your data with reports, analytics, and visualization.

- Build Reports and Dashboards Using Oracle Analytics
  You can use Oracle Analytics Cloud and Oracle Analytics Desktop with Autonomous Database.
- Creating Dashboards, Reports, and Notebooks with Autonomous Database
  Autonomous Database includes support for building dashboards, reports, and notebooks for data analysis using Oracle Analytics Desktop and Oracle Machine Learning Notebooks.
- Use Oracle Spatial with Autonomous Database
  Oracle Spatial Database is included in Autonomous Database, allowing developers and analysts to get started easily with location intelligence analytics and mapping services.
- The Data Analysis Tool
  The Data Analysis tool enables you to create Analytic Views with multidimensional metadata.
- The Catalog Page
  The Catalog page displays information about entities in the Oracle Autonomous Database.
- The Data Insights Page
  The Data Insights page displays information about patterns and anomalies in the data of entities in your Oracle Autonomous Database.
- Use Oracle OLAP with Autonomous Database
  Oracle OLAP is available on Autonomous Database. Oracle OLAP supports uses such as planning, forecasting, budgeting, and analysis.

Build Reports and Dashboards Using Oracle Analytics

You can use Oracle Analytics Cloud and Oracle Analytics Desktop with Autonomous Database.

Working with Oracle Analytics Cloud

You can use Oracle Analytics Cloud with Autonomous Database. Use Oracle Analytics Cloud to select interactive visualizations and automatically create advanced calculations to reveal the insights in your data.

For more information, see Use Oracle Analytics Cloud with Autonomous Database.

Working with Oracle Analytics Desktop

You can use Oracle Analytics Desktop with Autonomous Database. Just connect to Autonomous Database, select the elements that you're interested in, and let Oracle Analytics Desktop find the best way to visualize it. Choose from a variety of visualizations to look at data in a specific way.

For more information see Use Oracle Analytics Desktop with Autonomous Database.

Creating Dashboards, Reports, and Notebooks with Autonomous Database

Autonomous Database includes support for building dashboards, reports, and notebooks for data analysis using Oracle Analytics Desktop and Oracle Machine Learning Notebooks.

Oracle Analytics Desktop is designed for business users and report developers. Oracle Analytics Desktop enables you to gain insight into your data and create reports and build interactive visualizations. It also gives you a preview of the self-service visualization capabilities included in Oracle Analytics Cloud, which extends the data exploration and visualization experience by offering secure sharing and collaboration across the enterprise, additional data sources, and a full mobile experience including proactive self-learning analytics delivered to your device.

Oracle Machine Learning Notebooks is a built-in browser-based interactive data analysis environment that is part of Autonomous Database. Oracle Machine Learning Notebooks is designed for data scientists who want use the extensive library of in-database machine learning features.

- Create Dashboards and Reports to Analyze and Visualize Your Data
  Gain insight into your data with Oracle Analytics Desktop and Oracle Analytics Cloud. These tools let you explore your Autonomous Database data through advanced analytics and interactive visualizations.
- Create Notebooks, Workspaces, and Projects with Oracle Machine Learning Notebooks
  Oracle Machine Learning Notebooks provides a collaborative interface for creating notebooks and for performing real-time analytics.

Create Dashboards and Reports to Analyze and Visualize Your Data

Gain insight into your data with Oracle Analytics Desktop and Oracle Analytics Cloud. These tools let you explore your Autonomous Database data through advanced analytics and interactive visualizations.

- Use Oracle Analytics Desktop with Autonomous Database
  Gain insight into your data with Oracle Analytics Desktop. Oracle Analytics Desktop lets you explore your Autonomous Database data through interactive visualizations.
- Use Oracle Analytics Cloud with Autonomous Database
  You can use Oracle Analytics Cloud with Autonomous Database. Oracle Analytics Cloud provides a complete set of tools for deriving and sharing data insights.

Use Oracle Analytics Desktop with Autonomous Database

Gain insight into your data with Oracle Analytics Desktop. Oracle Analytics Desktop lets you explore your Autonomous Database data through interactive visualizations.

Oracle Analytics Desktop provides powerful personal data exploration and visualization in a simple per-user desktop download. Oracle Analytics Desktop is the perfect tool for quick exploration of sample data from multiple sources or for rapid analysis and investigation of your own local data sets.

Oracle Analytics Desktop makes it easy to visualize your Autonomous Database data so you can focus on exploring interesting data patterns. Just connect to Autonomous Database, select the elements that you're interested in, and let Oracle Analytics Desktop find the best way to visualize it. Choose from a variety of visualizations to look at data in a specific way.

Oracle Analytics Desktop also gives you a preview of the self-service visualization capabilities included in Oracle Analytics Cloud, Oracle's industrial-strength cloud analytics platform. Oracle Analytics Cloud extends the data exploration and visualization experience by offering secure sharing and collaboration across the enterprise, additional data sources, greater scale, and a full mobile experience including proactive self-learning analytics delivered to your device. Try Oracle Analytics Desktop for personal analytics and to sample a taste of Oracle's broader analytics portfolio.

Oracle Analytics Desktop's benefits include:

- A personal, single-user desktop application.
- Offline availability.
- Completely private analysis.
- Full control of data source connections.
- Lightweight single-file download.
- No remote server infrastructure.
- No administration tasks.

See the *User's Guide for Oracle Analytics Desktop* for information on connecting Oracle Analytics Desktop to Autonomous Database.

To get started with Oracle Analytics Desktop use the following link to visit the software download page which includes more information about system requirements and provides instructions for installing Oracle Analytics Desktop on different platforms:

Oracle Analytics Desktop Download Details

Use Oracle Analytics Cloud with Autonomous Database

You can use Oracle Analytics Cloud with Autonomous Database. Oracle Analytics Cloud provides a complete set of tools for deriving and sharing data insights.

- Data preparation: Analysts can ingest, profile, and cleanse data using a variety of algorithms.
- Data flow: Analysts can prepare, transform and aggregate data, and then run machine-learning models at scale.
- Data discovery: Subject matter experts can easily collaborate with other business users, blending intelligent analysis at scale, machine learning, and statistical modeling.
- Data visualization: Analysts can visualize any data, on any device, on-premise and in the cloud.
- Data collaboration: Large organizations and small teams can share data more simply, as you don't need to manage or consolidate multiple versions of spreadsheets.
- Data-driven: Application developers can utilize interfaces that enable them to extend, customize, and embed rich analytic experiences in the application flow.

See *Visualizing Data and Building Reports in Oracle Analytics Cloud* for details on connecting Autonomous Database with Oracle Analytics Cloud.

Create Notebooks, Workspaces, and Projects with Oracle Machine Learning Notebooks

Oracle Machine Learning Notebooks provides a collaborative interface for creating notebooks and for performing real-time analytics.

- Using Oracle Machine Learning User Interface on Autonomous Database
  Oracle Machine Learning Notebooks provide a browser-based interactive data analysis environment where you can develop, document, share, and automate analytical methodologies.

- Work with Oracle Machine Learning User Interface for Data Access, Analysis, and Discovery
  You can use Oracle Machine Learning Notebooks to access data and for data discovery, analytics, and notebooks.

Using Oracle Machine Learning User Interface on Autonomous Database

Oracle Machine Learning Notebooks provide a browser-based interactive data analysis environment where you can develop, document, share, and automate analytical methodologies.

Oracle Machine Learning Notebooks includes:

- **Oracle Machine Learning User Administration Application**
  - Web based administrative UI for managing (list, create, update, delete) Oracle Machine Learning users
  - Notebook users map to database users.
  - Access to the User Management feature is limited to the database administrator account (ADMIN). See Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database for more information.
- **Oracle Machine Learning Notebooks Application**
  - Web based application for data scientists
  - Allows for creation of workspaces, projects, and notebooks

  See Work with Oracle Machine Learning User Interface for Data Access, Analysis, and Discovery for information on accessing Oracle Machine Learning user workspaces, projects, and notebooks from your Autonomous Database.

Work with Oracle Machine Learning User Interface for Data Access, Analysis, and Discovery

You can use Oracle Machine Learning Notebooks to access data and for data discovery, analytics, and notebooks.

To access Oracle Machine Learning Notebooks you can use the Oracle Cloud Infrastructure Console or Database Actions.

To access Oracle Machine Learning Notebooks from the Oracle Cloud Infrastructure Console:

1. From the Display Name column, select an Autonomous Database.
2. On the Autonomous Database Details page click the **Tools** tab.
3. In the **Tools** column, select the **Oracle ML User Administration** link.
4. Enter your **Username** and **Password**.

   The Administrator creates OML users. See Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database for details.
5. Click **Sign in**.

To access Oracle Machine Learning Notebooks from Database Actions:

1. On the Oracle Cloud Infrastructure Console, from the Display Name column select an Autonomous Database.
2. On the Autonomous Database Details page select **Database Actions** and click **View all database actions**.

3. On the Database Actions Launchpad, under **Development**, select **Oracle Machine Learning**.

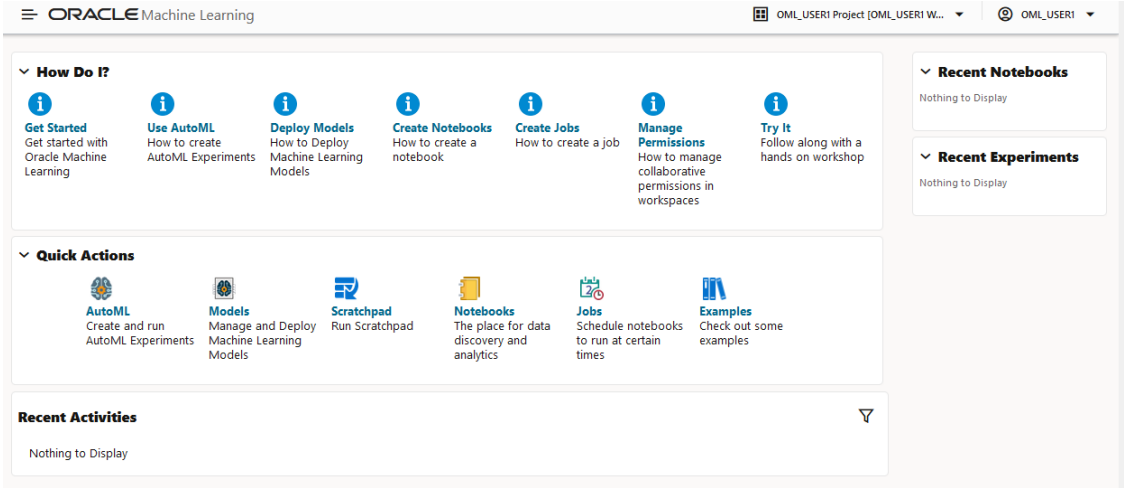
   If you do not see the **Oracle Machine Learning** card then your database user is missing the required `OML_DEVELOPER` role. See Required Roles to Access Tools from Database Actions for more information.

4. Enter your username and password.

   The Administrator creates OML users. See Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database for details.

5. Click **Sign In**.

This shows Oracle Machine Learning user application.



Oracle Machine Learning Notebooks allows you to access your data in your database and build notebooks with the following:

- Data Ingestion and Selection
- Data Viewing and Discovery
- Data Graphing, Visualization, and Collaboration
- Data Analysis

You can also create and run SQL statements and create and run SQL scripts that access your data in your database.

Use Oracle Spatial with Autonomous Database

Oracle Spatial Database is included in Autonomous Database, allowing developers and analysts to get started easily with location intelligence analytics and mapping services.

- About Oracle Spatial with Autonomous Database
- Oracle Spatial Limitations with Autonomous Database
  Autonomous Database includes Oracle Spatial, with some limitations.

About Oracle Spatial with Autonomous Database

Oracle Spatial allows developers and analysts to use spatial analysis in every application from basic spatial search and analysis to advanced geospatial applications and Geographic Information Systems (GIS).

Organizations can manage different types of geospatial data, perform hundreds of spatial analytic operations, and use interactive map visualization tools with the spatial features in Oracle Autonomous Database and Oracle Database.

The spatial features provide a schema and functions that facilitate the storage, retrieval, update, and query of collections of spatial features in Autonomous Database. Spatial consists of the following:

- A schema (MDSYS) that prescribes the storage, syntax, and semantics of supported geometric data types
- A spatial indexing mechanism
- Operators, functions, and procedures for performing area-of-interest queries, spatial join queries, and other spatial analysis operations
- Utility functions and procedures for validating, loading, extracting, and working with spatial data
- GeoRaster, a feature that lets you store, index, query, analyze, and deliver GeoRaster data, that is, raster image and gridded data and its associated metadata
- Features for geocoding address data and for reverse geocoding longitude/latitude data to a street address:
  - SDO_GCDR.ELOC_GEOCODE
  - SDO_GCDR.ELOC_GEOCODE_AS_GEOM

> **Note:**
>
> `SDO_GCDR.ELOC_GEOCODE` and `SDO_GCDR.ELOC_GEOCODE_AS_GEOM` functions are only available on Autonomous Database.

See the following for more information:

- Spatial Concepts
- Oracle Database Spatial
- Geocoding Address Data

Oracle Spatial Limitations with Autonomous Database

Autonomous Database includes Oracle Spatial, with some limitations.

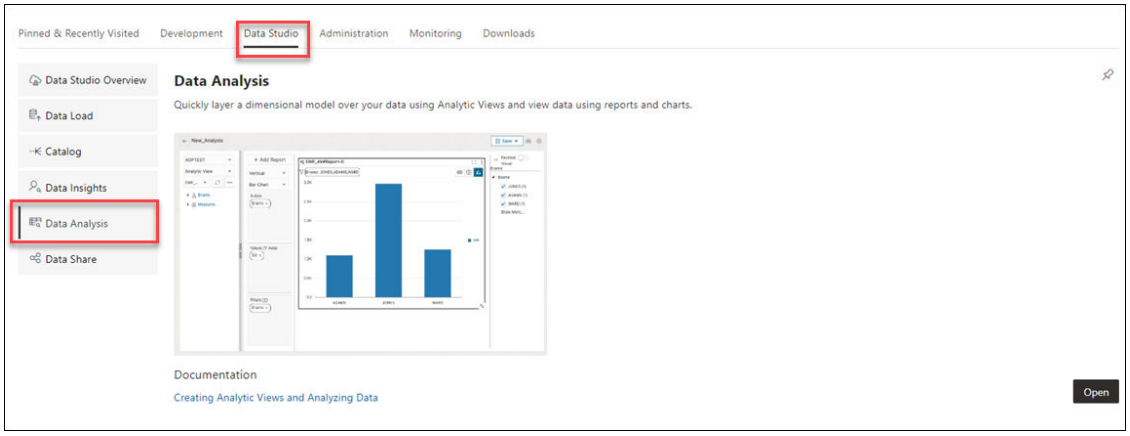
You can use the Oracle spatial features with Autonomous Database, with the following limitations:

- The following Oracle Spatial components require that you download applications from Oracle Cloud Marketplace:
  - Oracle Spatial Studio
  - Oracle Spatial Map Visualization
  - Network Data Model
  - Geospatial Consortium Web Services
- The following is not available in Autonomous Database:
  - Router

The Data Analysis Tool

The Data Analysis tool enables you to create Analytic Views with multidimensional metadata.

To reach the **Data Analysis** page, select the **Data Analysis** menu in the Data Studio tab of the Launchpad.

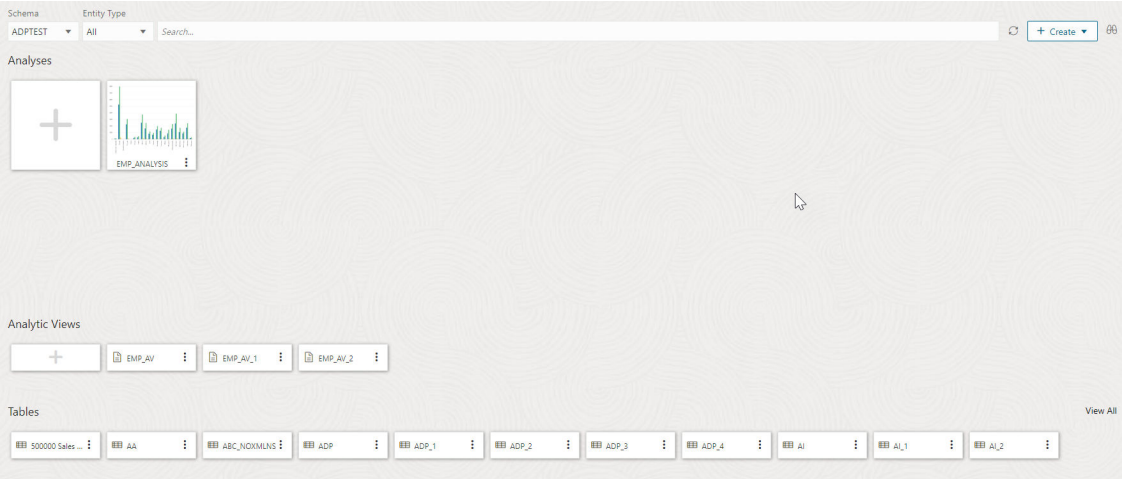


You create Analytic Views on top of a fact table with several dimensions and hierarchies. Analytic views refer to tables in the database and allow users to create hierarchies for dimensions. You can also create Analyses and reports using information from the Analytic Views. The Data Analysis homepage enables you to search for Analyses, view and perform tasks such as edit, delete, view or rename Analyses. You can also analyze, find errors, export, edit, compile and delete Analytic Views. You can analyze tables and generate SQL reports from them.

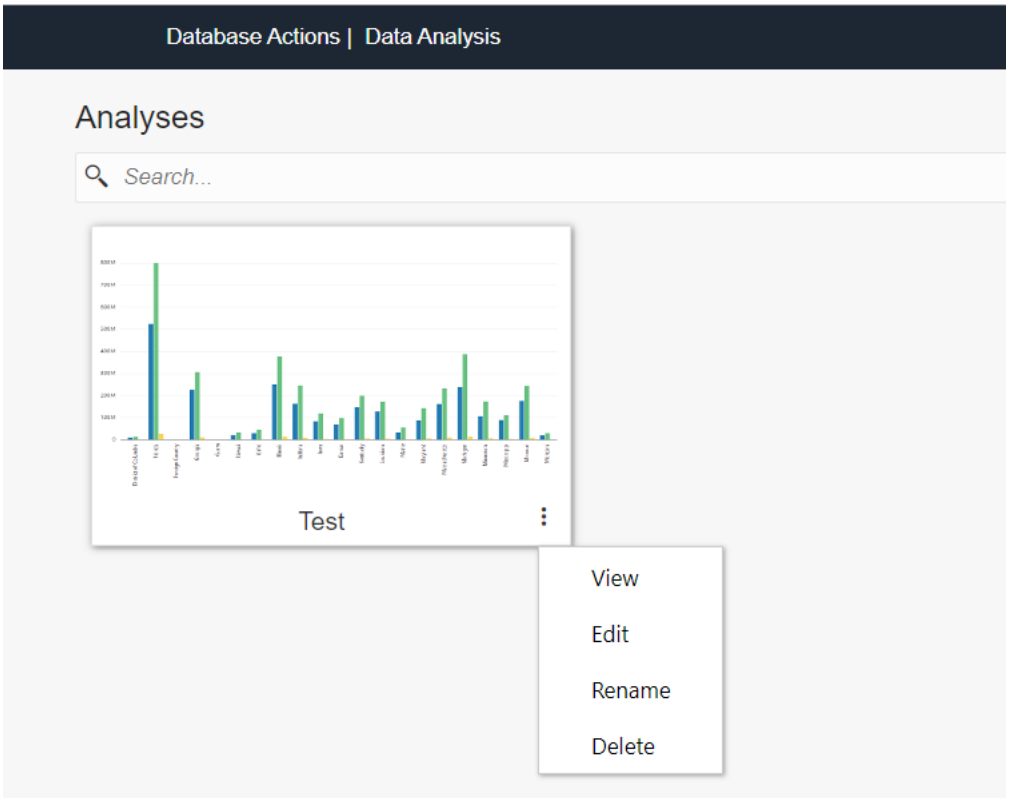
Select the **Data Analysis** card from the Data Studio suite to access this tool. You can also access it by clicking the Selector icon and selecting Data Analysis from the Data Tools menu in the navigation pane.

> **Note:**
>
> If you do not see the Data Analysis card then your database user is missing the required DWROLE role.

The Data Analysis home page consists of three parts: Analyses, Analytic Views and Tables.



Analyses



The top section of the homepage comprises of a list of Analyses. Use the search field to search for Analyses you create. The top section of the homepage comprises of a list of Analyses. Use the search field to search for Analyses you create.

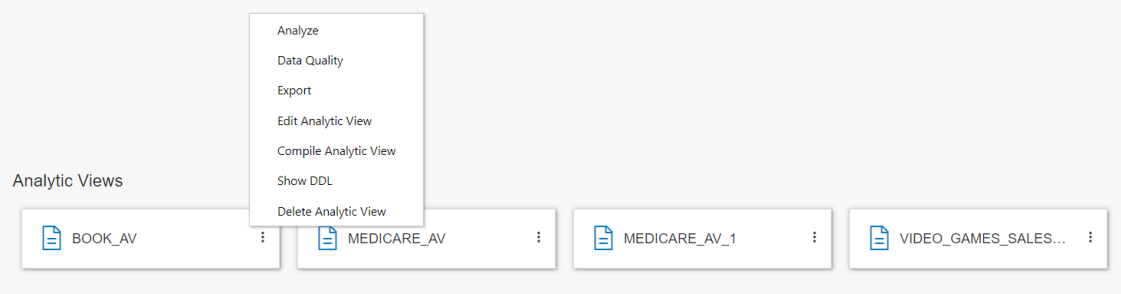
Analyses are analysis of multiple Analytic Views. The Analyses card displays the name of the analysis. Click **Actions** (three vertical dots) to open the context menu.

The actions available are:

- **View:** Opens the Analysis View page in a new window where you can view the analysis.
- **Edit:** Opens the selected Analysis page where you can edit the reports present in the analysis.

- **Rename:** Allows you to rename the Analysis. Click **save** to modify the new name.
- **Delete:** Opens the delete Analysis dialog where you can delete the analysis.

Analytic Views



The bottom section of the homepage displays list of existing Analytic Views. Each Analytic View card displays the name of the Analytic View. The Actions icon enables you to manage the Analytic View. Click **Actions** (three vertical dots) to open the context menu. The actions available are:

- **Analyze:** Opens the Analytic View browser and the Analysis View page in a new window where you can view the Analysis.
- **Data Quality:** Opens the Data Quality page where the tool validates the selected Analytic View for errors and lists them out.
- **Export:** Allows you to export the Analytic View to Tableau and PowerBI.
- **Edit Analytic View:** Opens the Edit Analytic View dialog box where you can edit the properties of the selected Analytic View.
- **Compile Analytic View:** This option compiles the Analytic View and returns compilation errors if there are any.
- **Show DDL:** Displays the DDL statements for the Analytic View.
- **Delete Analytic View:** Deletes the selected Analytic View.

The **+Create** button enables you to create **Analysis** and create **Analytic View** from the home page.

You can select both hierarchies and measures from Analytic Views. Hierarchies are DB objects that allow users to define relationships between various levels or generations of dimension members. As the name implies, hierarchies organize data using hierarchical relationships. With this tool you can analyze and visualize data in different Points of View (POV). You can export the metadata and visualize it with sophisticated tools like Oracle Analytics Cloud (OAC) and Tableau.

Advantages of Data Analysis tool

With Data Analysis tool you can:

- Visualize, analyze and inspect your data clearly and efficiently with pivot tables
- Calculate total number of errors present in the Analytic View you create and provide solutions to minimize the errors
- Automatically display meaningful insights to help you make better decisions
- Analyze your data across dimensions with support for hierarchical aggregation and drill-down
- Share your Analytic Views with the tool of your choice over various options of raw data consumption to draw meaningful insights and make them accessible to any user

By identifying relationships among tables and columns, Analytic Views enable your system to optimize queries. They also open new avenues for analyzing data. These avenues include data insights, improved hierarchy navigation, and the addition of hierarchy-aware calculations.

This tool runs complex and hierarchical SQL queries along with SQL extensions in the background, which simplifies real-time calculations. It makes complex data more accessible and easier to understand.

The Data Analysis Page

The following section describes searching and obtaining information about Analytic Views, creating Analytic Views, inspecting your data, discovering insights and visualizing data using tools like Oracle Analytics Cloud (OAC), Tableau, and Microsoft Power BI.

> **Note:**
>
> - OAC has in-built tools to search and utilize Analytic Views.
> - We have no direct support for Microsoft Power BI, yet its users can map their tool to the AV transparency views to avail some of the benefits of Analytic Views.

Read these topics for detailed descriptions on Analyses and Analytic Views:

- Searching and obtaining information about Analytic Views
- Creating Analytic Views
- Working with Analyses
- Viewing Analyses
- Workflow to build analyses
- Creating Analyses
  - Creating Report
  - Saving Analyses
- Searching and obtaining information about Analytic Views
  When you first open the Data Analysis page, it displays the list of schemas and Analytic Views. With Select Schema, you can select a preferred Schema from a list of schemas available in the drop-down.
- Creating Analytic Views
  You can create Analytic Views and view information about them. You can also edit and perform other actions on them.
- Working with Analyses
  Analyses are a collection of multiple reports on a single page, which provides quick access to multiple data analyses collected from different Analytic Views.
- Viewing Analyses
  Analyses provide you an insight into the performance of your data.
- Workflow to build Analyses
  Here is the workflow to build an analyses.

- Creating Analyses
  Use the Data Analysis tool to create and edit your Analyses. The analysis provides you customized view of Analytic View data. An analysis consists of one or more reports that displays the results of analysis.
- Creating Reports
  A single report you generate analyzes an AV based on the Levels and measures you select.
- Using Calculation Templates
  The Data Analysis tool provides templates for all of the calculations typically in demand for business intelligence applications.
- Oracle Autonomous Database Add-in for Excel
  The Oracle Autonomous Database Add-in for Excel integrates Microsoft Excel spreadsheets with the Autonomous Database to retrieve and analyze data from Analytic Views in the database. You can also directly run SQL queries to view their results in the worksheet.
- Oracle Autonomous Database add-on for Google Sheets
  The Oracle Autonomous Database add-on for Google Sheets enables you to query tables using SQL or Analytic Views using a wizard directly from Google Sheets for analysis.

Searching and obtaining information about Analytic Views

When you first open the Data Analysis page, it displays the list of schemas and Analytic Views. With Select Schema, you can select a preferred Schema from a list of schemas available in the drop-down.

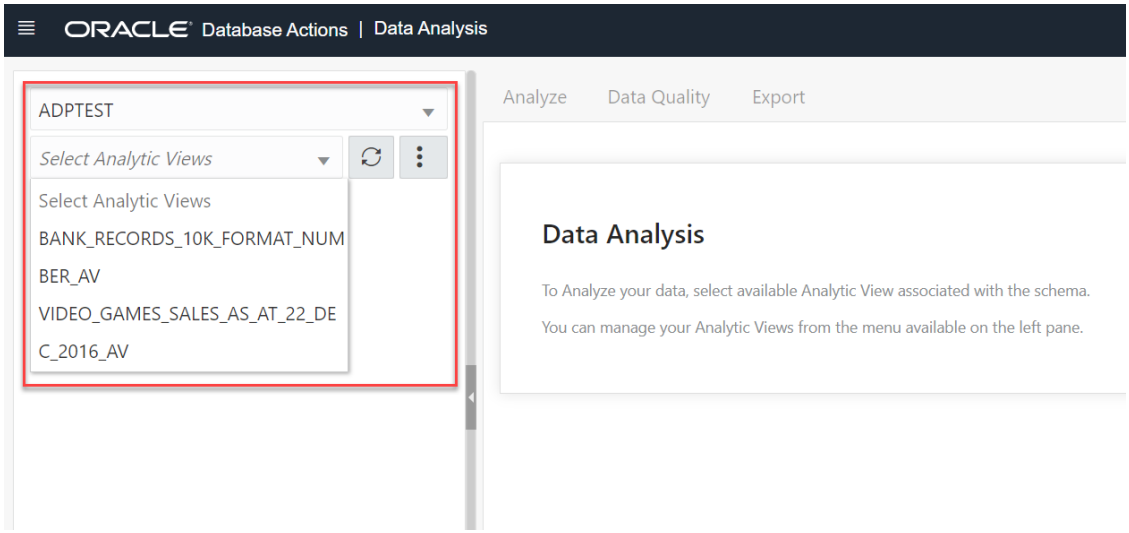
The **Select Analytic Views** drop-down enables you to select an available Analytic View associated with the schema. When you create an Analytic View, it appears in the drop-down option with your schema. The Refresh AV icon refreshes the contents of the selected Analytic View.

The **Action** icon next to the Refresh AV button enables you to manage Analytic Views. You can Create Analytic View, Edit Analytic View, Compile Analytic View, Show the Data Definition Language (DDL) that generates the Analytic View or Delete Analytic View from the menu.

Obtain information about Analytic Views

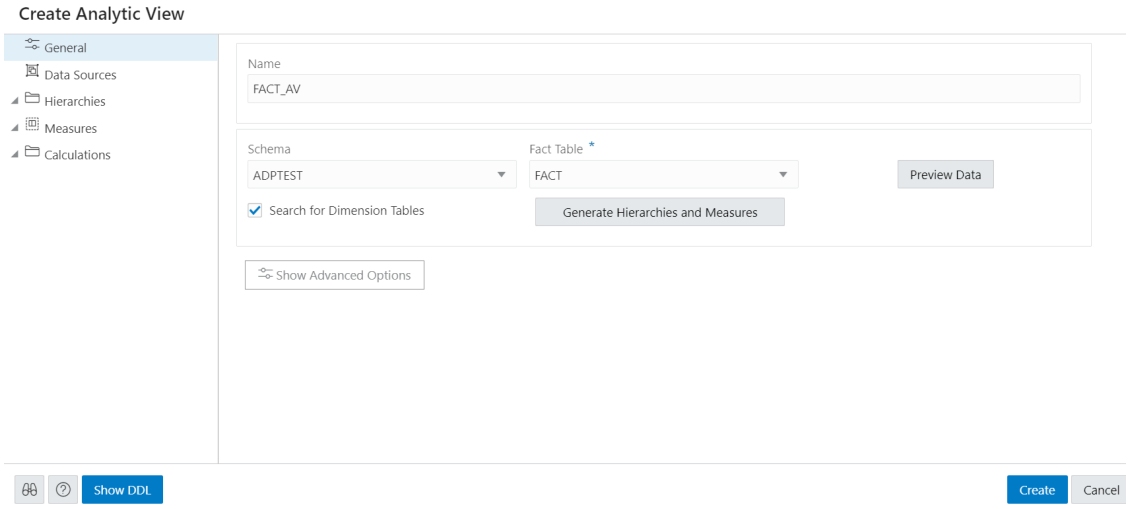
By default, Analytic Views are filtered by the current user's schema, as indicated by the schema list below the menu-bar. You can remove the selected schema filter by selecting another user's schema. To search for Analytic Views in other schemas, select one of the schemas from the drop-down.

If there is no Analytic View associated with the schema selected, the tool prompts you to create an Analytic View.



Creating Analytic Views

You can create Analytic Views and view information about them. You can also edit and perform other actions on them.



When you create an Analytic View, you identify a fact table that contains the data to inspect. The **Generate Hierarchies and Measures** button looks at the contents of that table, identifies any hierarchies in the fact table, and searches for other tables that may contain related hierarchies.

While creating an Analytic View, you can enable or disable the following advanced options:

- Autonomous Aggregate Cache, which uses the dimensional metadata of the Analytic View to manage a cache and that improves query response time.
- Analytic View Transparency Views, which presents Analytic Views as regular database views and enables you to use your analytic tools of choice while gaining the benefits of Analytic Views.
- Analytic View Base Table Query Transformation, which enables you to use your existing reports and tools without requiring changes to them.

Create Analytic View

To create Analytic View, click **Create** from the Data Analysis home page and select **Create Analytic View** to begin the process.

Click **Cancel** to cancel the creation of the Analytic View at any time.

Specify Attributes of the Analytic View

On the **General** tab of the **Create Analytic View** pane, specify the following:

- The name for the Analytic View
- The fact table for the view
- Advanced options

You can also preview the data of the fact table and see statistics about that data.

In the **Name** field, specify a name of your choice.

The **Schema** field has the current user's schema. You can only create an Analytic View in that schema.

In the **Fact Table** field, expand the drop-down list and click **More Sources**. The **Select Sources** dialog box has a list of the available tables and views. Select a table or view from the list.

To filter the list, begin typing characters in the Filter field. As you type, the list changes to show the tables or views that contain the characters. Clear the field to show the complete list again. After you select a table or view, click OK.

To enable or disable the advanced options, on the **Create Analytic View** pane, click the **Show Advanced Options** icon at the bottom left. Select or deselect options as desired.

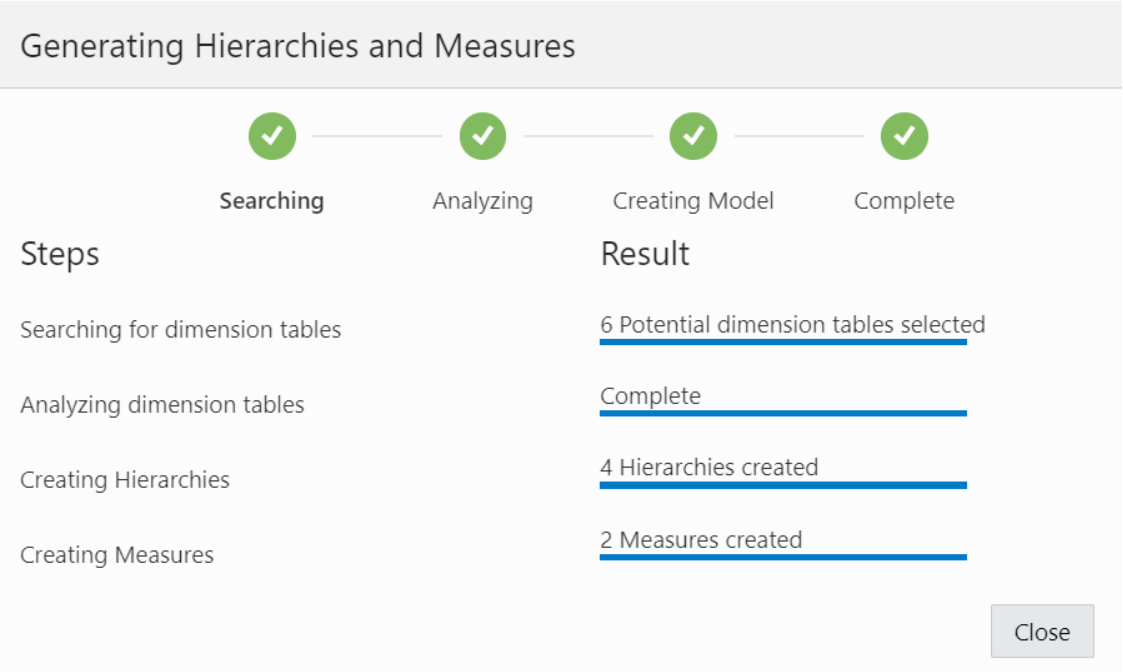
To view the data in the fact table and statistics about the data, click the **Preview Data** button. In the Preview and Statistics pane, the Preview tab shows the columns of the table and the data in the columns.

The Statistics tab shows the size of the table and the number of rows and columns. The statistics may take a few moments to appear, during which time the message, "No statistics available..." may appear. The statistics include the names of the columns, their data types, the number of distinct values and null values, the maximum and minimum values, and other information. The bar graph displays the top unique column values and the number of their occurrences for the selected column. Point to a bar in the graph to see the number of occurrences of the unique value.

Click **Close** to close the Preview and Statistics pane and return to the Create Analytic View pane.

Click on **Generate Hierarchies and Measures** icon.

The Generating Hierarchies and Measures dialog box displays the progress of searching for dimension tables, analyzing the dimension tables and identifying and creating the data sources, joins, hierarchies, and measures to use. When the process completes, click **Close**.



The **Search for Dimension Tables** check box when selected, enables you to search for dimension tables while generating hierarchies and measures.

After the hierarchies and measures are generated, they are displayed under their respective tabs. Review the hierarchies and measures created for you.

Specify the **Name**, **Fact Table** and select **Advanced Options** in the **General** tab of **Create Analytic View** pane. Click **Create** to generate an Analytic View.

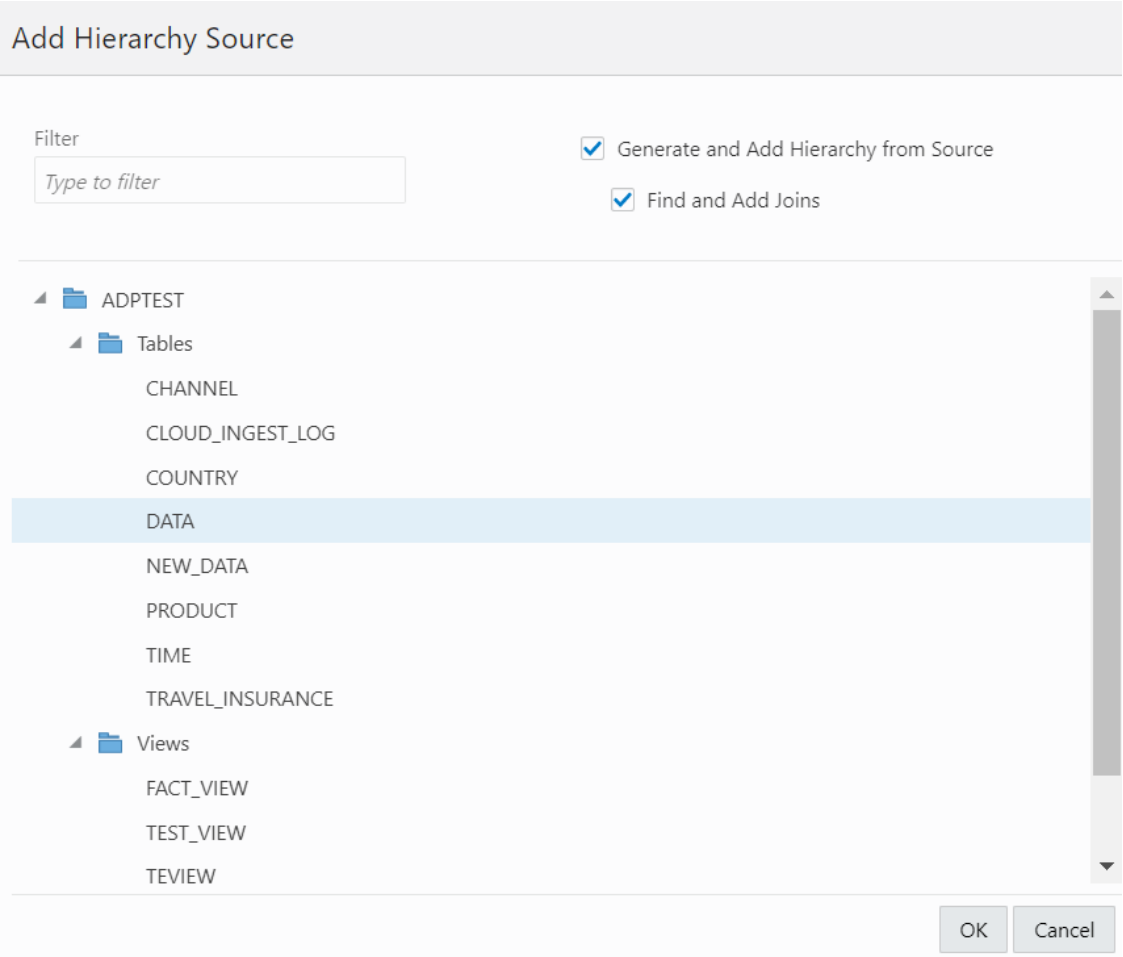
View Data Sources

The Data Sources tab displays the sources of the data and the relationships among them. It has a graphical display of the fact table and the related dimension tables. For example, a fact table of health insurance data might have columns for geography identifiers, income codes, and gender codes. The Data Sources tab would display items for the fact table and for the geography, income, and gender dimension tables.

You can add hierarchies from data sources even after generating hierarchies from the existing fact table. You can add one or more hierarchies to your new or existing analytic view. Multiple hierarchies can be defined and used in an analytical view, however only one will be used by default.

Right-click the Data Sources tab and select **Add Hierarchy Sources** or select **Add Hierarchy Sources**.

Selecting **Add Hierarchy Sources** launches an **Add Hierarchy Source** dialog box.



You can view all the fact tables and views associated with the analytic view.

In the filter field, you can either manually look for the source or start typing to search for the fact table or views from the list of available fact tables and views. After typing the full name of the source, the tool automatically matches the fact table or view.

Select **Generate and Add hierarchy from Source** to generate analysis and hierarchies associated with the source data you select.

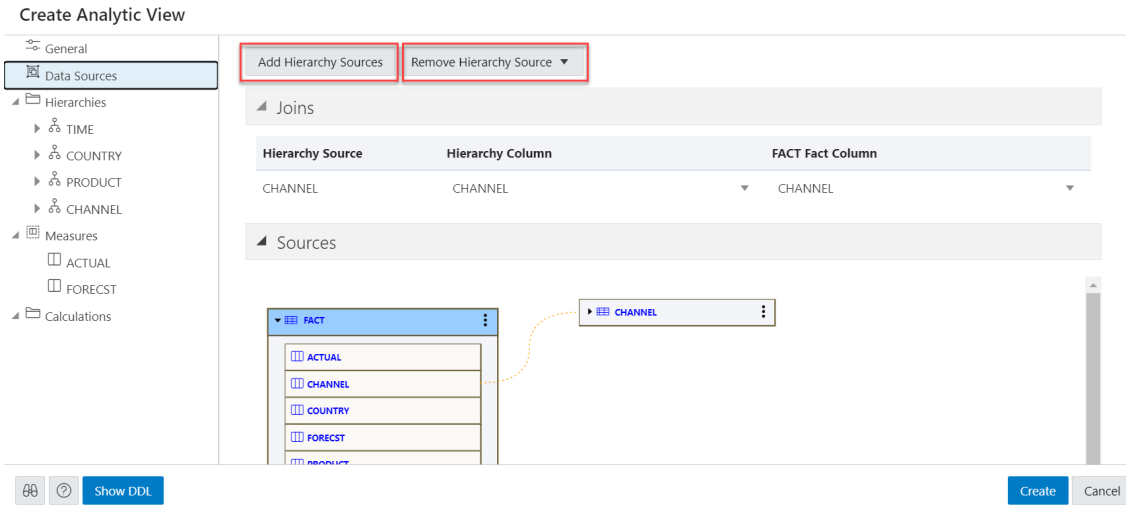
Select **Find and Add Joins** to link all the data sources with the fact table. You can add multiple join entries for a single hierarchy.

Click **OK** to select the source.

The Generating Hierarchies and Measures dialog box displays the progress of analyzing the dimension tables and creating the hierarchies. When the process completes, click **Close**.

> **Note:**
>
> When you add a hierarchy from the data source, you see the new hierarchy in the list of hierarchies in the Hierarchies tab. You can navigate between the Data Sources tab, the Hierarchies tab, the Measures tab, the Calculations tab. You can add a hierarchy from a source that is not connected by navigating back to the Data Sources tab.

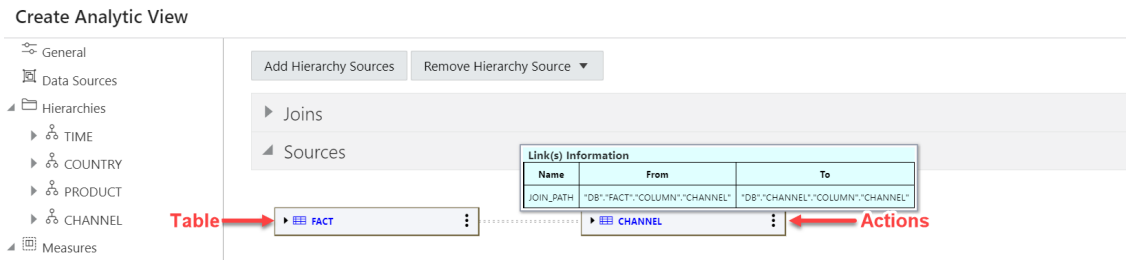
Select **Remove Hierarchy Source** to remove the hierarchies you create from the data sources. You cannot remove hierarchies generated from the fact table you select from this option.



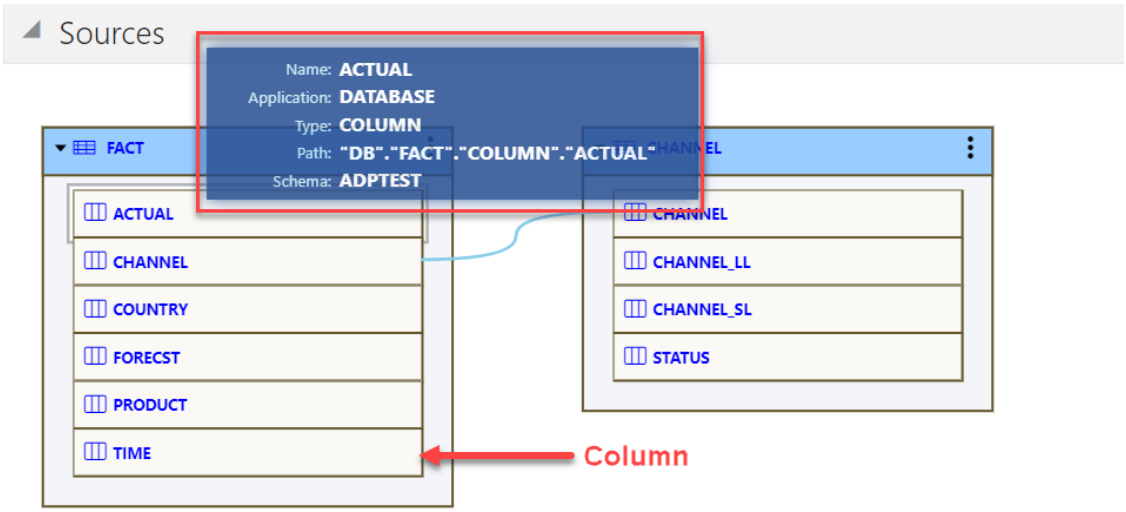
Expand **Joins** to view the **Hierarchy Source**, **Hierarchy Column** and the **Fact column** mapped with the Analytic View. The **Joins** is visible only when the hierarchy table differs from the fact table. You can add multiple join entries for a single hierarchy.

Expand **Sources** to view the fact table associated with the Analytic View. The data model expands to include the data from the source that you added.

Pointing to an item displays the name, application, type, path and the schema of the table. Click the **Actions** (three vertical dots) icon at the right of the item to display a menu to expand or collapse the view of the table.



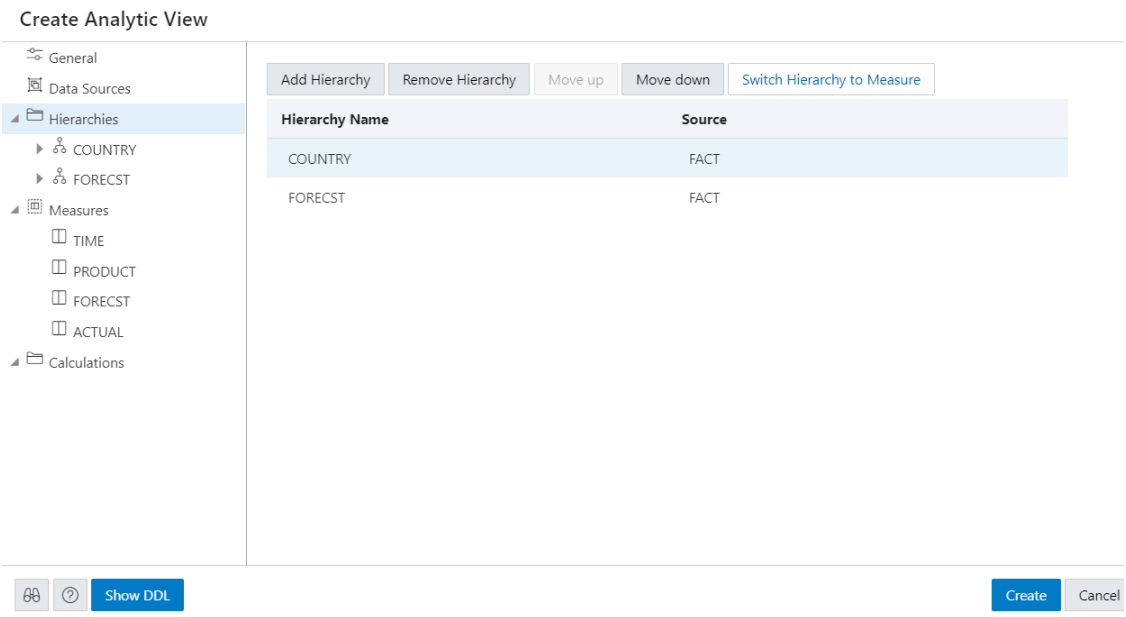
An expanded item displays the columns of the table. Pointing to a column displays the name, application, type, path, and schema of the column.



The lines that connect the dimension tables to the fact table indicate the join paths between them. Pointing to a line displays information about the join paths of the links between the tables. If the line connects a table that is collapsed, then the line is dotted. If the line connects two expanded tables, then the line is solid and connects the column in the dimension table to the column in the fact table.

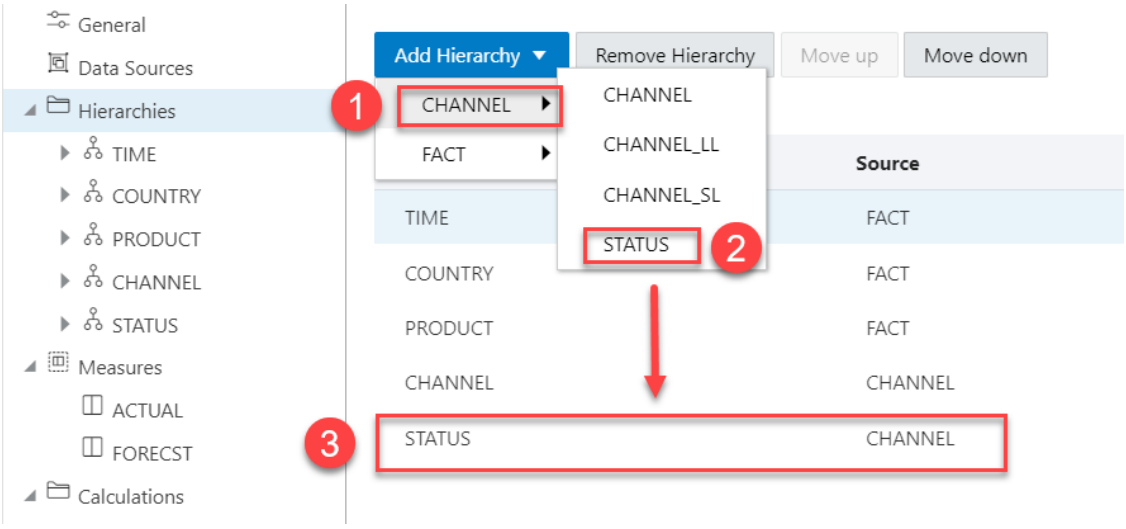
View and Manage Hierarchies

The Hierarchies tab displays the hierarchies generated by the Analytic View creation tool. The display includes the name of the hierarchy and the source table.



An analytic view must include at least one hierarchy.

To add a Hierarchy, click **Add Hierarchy**. This results in a display as a list of column in that table. Select a column that operates as the detailed level of the hierarchy and be the join-key to the fact table.



To remove the hierarchy, select the hierarchy you want to remove from the list and click **Remove Hierarchy**

Select **Move Up** or **Move Down** to position the order of the Hierarchy in the resulting view.

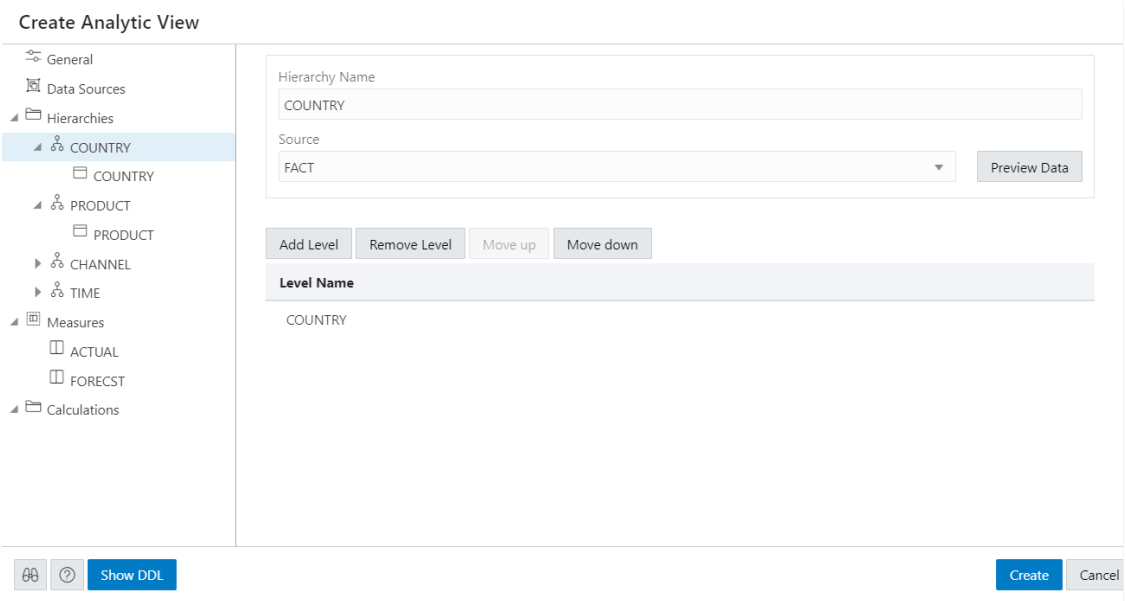
Click **Switch Hierarchy to Measure** to change the hierarchy you select to a measure in the Measures list.

You can also **Add Hierarchy** and **Add Hierarchy From Table** by right-clicking the Hierarchy tab.

If you click on a hierarchy name, a dialog box displays the Hierarchy Name and Source.

To change the source, select a different source from the drop-down list.

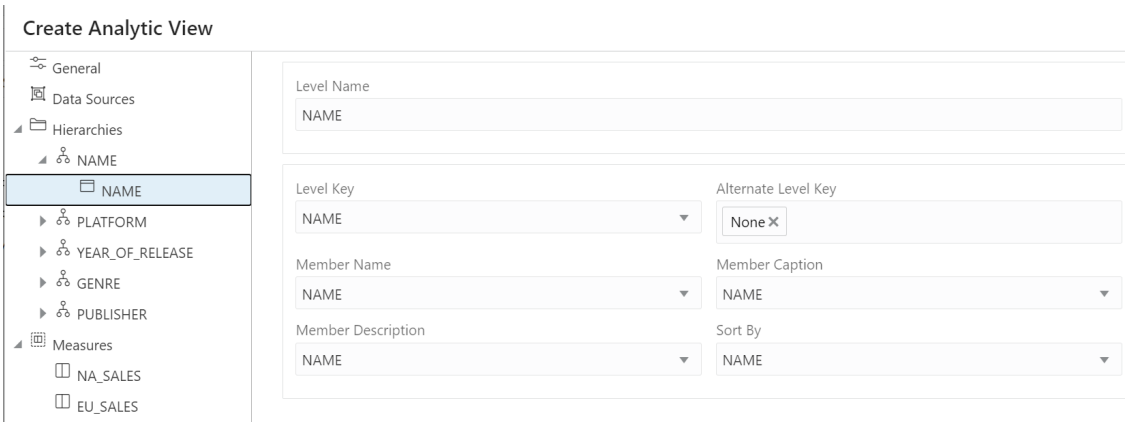
Select **Add Level** to add a level to the hierarchy. Click **Remove Level** to remove the selected level from the hierarchy.



To view the data in the fact table and statistics about the data, click the **Preview Data** button. In the Preview and Statistics pane, the Preview tab shows the columns of the table and the data in the columns. The Statistics tab shows the size of the table and the number of rows and columns.

If you click on a particular level in the Hierarchy tab, a dialog box displays it's respective Level Name, Level Key, Alternate Level Key, Member Name, Member Caption, Member Description, source, and Sort By drop-down. To change any of the field values, enter the value in the appropriate field.

> **Note:**
>
> You can enter multiple level keys Member Name, Member Caption, Member Description and Sort By.



Member Captions and Member Descriptions generally represent detailed labels for objects. These are typically end-user-friendly names. For example, you can caption a hierarchy representing geography areas named GEOGRAPHY_HIERARCHY as "Geography" and specify its description as "Geographic areas such as cities, states, and countries."

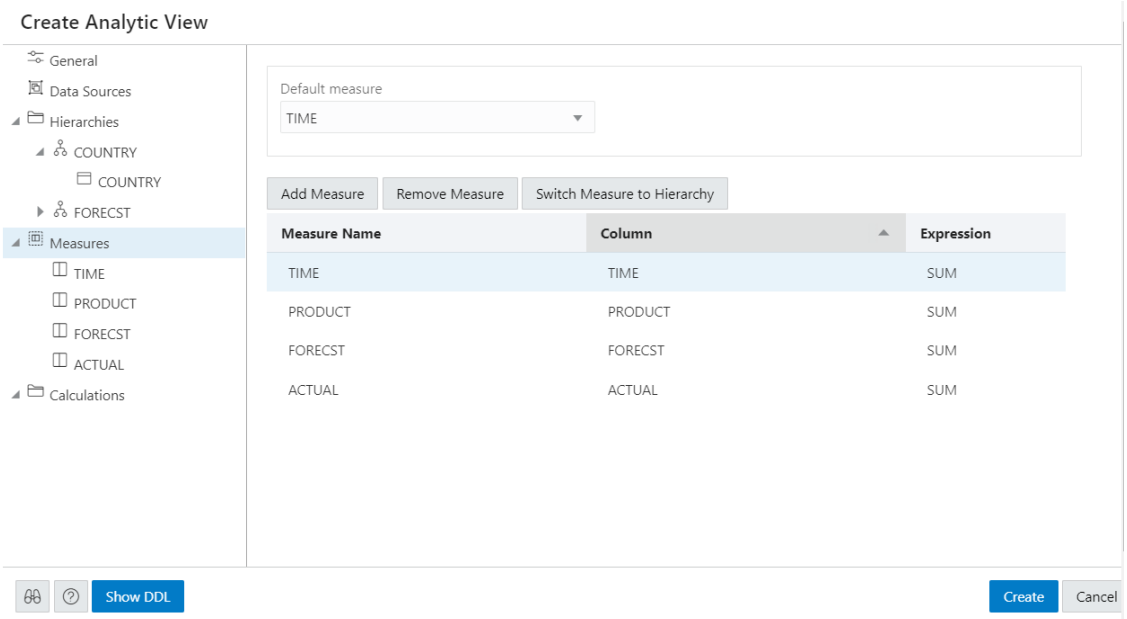
To see the measures for the Analytic View, click **Measures** tab. To immediately create the Analytic View, click **Create**. To cancel the creation, click **Cancel**.

View and Manage Measures

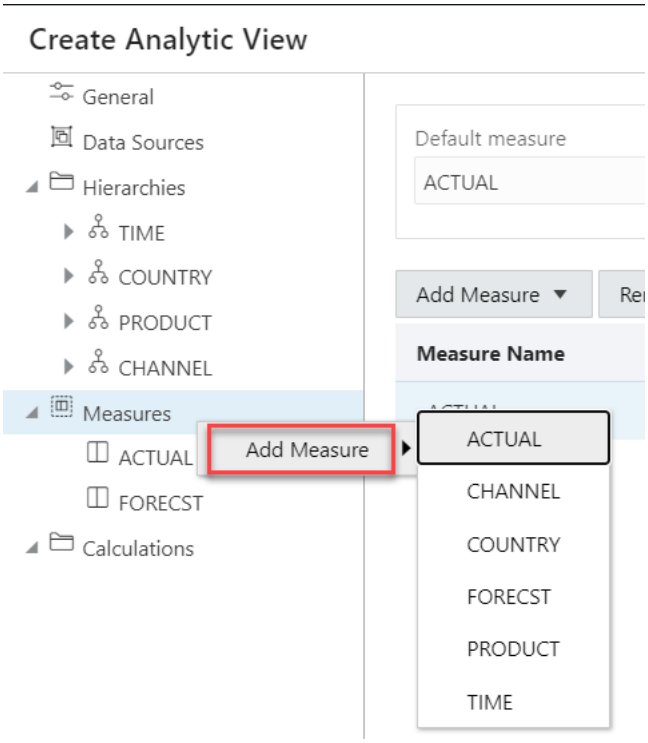
The Measures tab displays the measures suggested for the Analytic View. It displays the Measure Name, Column, and operator Expression for each measure.

The measures specify fact data and the calculations or other operations to perform on the data.

To add measures, click **Add Measure**. You can view a new measure at the bottom of the measures list. To remove the measure, select the measure you want to remove from the list and click **Remove Measure**.



To alternatively add a measure from the data source, right- click the Measures tab. This pops up a list of columns that can be used as measures. Select one measure from the list.



You can exclude a column from the measures on right-clicking the Measures tab and selecting Remove Measure.

Click **Switch Measure to Hierarchy** to change the measure you select to hierarchy in the Hierarchies list.

You must specify a measure as the default measure for the analytic view; otherwise, the first measure in the definition is the default. Select **Default Measure** from the drop-down.

To add a measure, right-click the Measures tab and select **Add Measure**. To remove a measure, select the particular measure you want to remove, right-click on it and select **Remove Measure**.

You can select a different column for a measure from the Column drop-down list. You can select a different operator from the Expression drop-down list.

In creating an analytic view, you must specify one or more hierarchies and a fact table that has at least one measure column and a column to join to each of the dimension tables outside of the fact table.

Create new calculated measures

You can add measure calculations to a query of an analytic view.

The measures and hierarchies associated with the analytic views enable us to create new calculated measures.

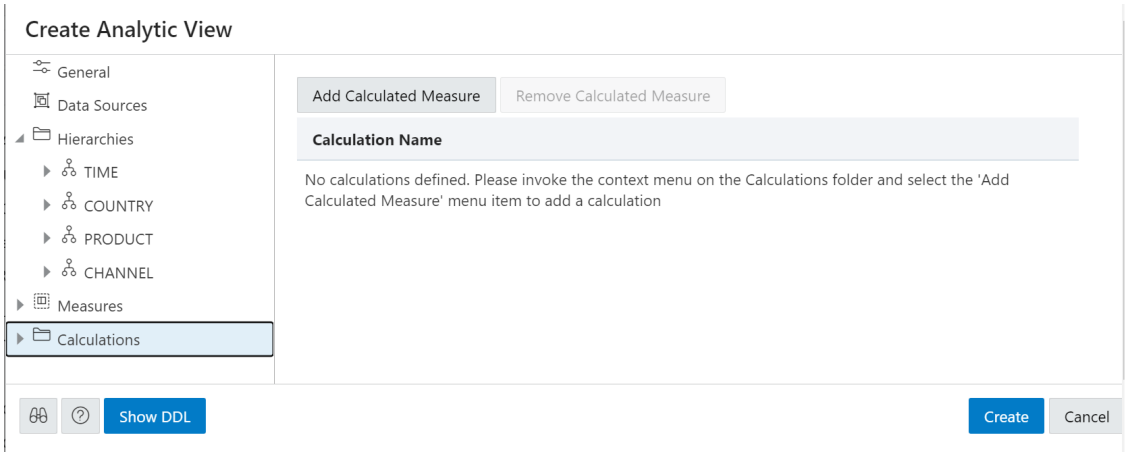
Calculated measures return values from data stored in one or more measures. You compute these measures at run time.

> **Note:**
>
> You can create the measures without increasing the size of the database since the calculated measures do not store the data. However, they may slow performance. You need to decide which measures to calculate on demand.

The Analytic Views provides easy-to-use templates for creating calculated measures.

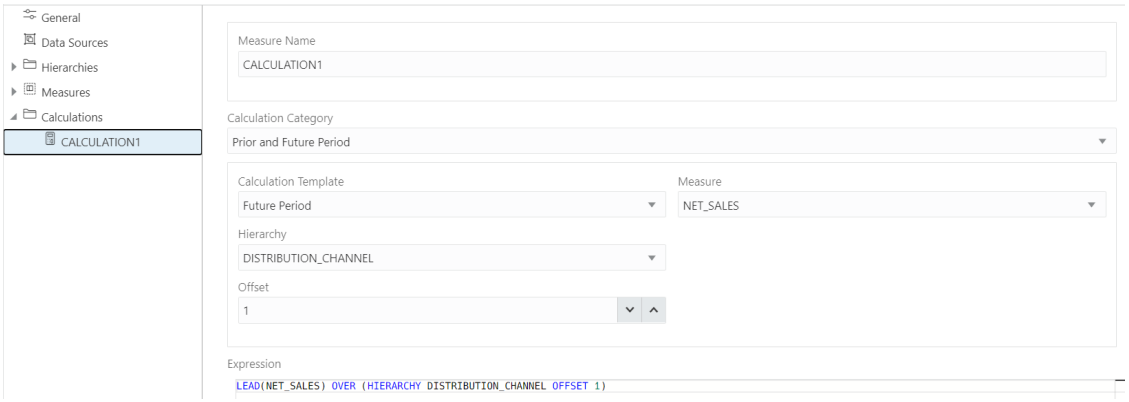
Once you create a calculated measure, it appears in the list of measures of the Analytic View .You can create a calculated measure at any time which is available for querying in SQL.

The Data Analysis tool provides easy-to-use templates for creating calculated measures.



Click **Add Calculated Measure** to add calculations to the measures. You can view the new calculation with system generated name in the **Calculations** tab.

Click the newly created calculated measure.



In the **Measure Name** field, enter the name of the calculated measure.

You can select preferred category of calculation from a list of options such as Prior and Future Period, Cumulative Aggregates, Period To Date, Parallel Period, Moving Aggregates, Share, Qualified Data Reference, and Ranking using the **Calculation Category** drop-down.

Your choice of category of calculation dynamically changes the **Calculation Template**.

For more details on how to use Calculation templates, see Using Calculation Templates.

Select the **Measure** and **Hierarchy** on which you want to base the calculated measures.

Select **Offset** value by clicking the up or the down arrow. The number specifies the number of members to move either forward or backward from the current member. The ordering of members within a level is dependent on the definition of the attribute dimension used by the hierarchy. The default value is 0 which represents POSITION FROM BEGINNING.

The Expression field lists the expressions which the calculated measure uses.

On the creation of the Analytic view, the calculated measure appears in the navigation tree in the Calculated Measures folder.

Click **Create**. A confirmation dialog box appears that asks for your confirmation. Select **Yes** to proceed with the creation of Analytic View.

After creating the Analytic View, you will view a success message informing you of its creation.

On editing the Analytic View you create, you can view the calculated measure in the navigation tree in the Calculations folder.

Click the **Tour** icon for a guided tour of the worksheet highlighting salient features and providing information if you are new to the interface.

Click the **help** icon to open the contextual or online help for the page you are viewing.

Click **Show DDL** to generate Data Definition Language statements for the analytic view.

Edit Analytic View

You might want to edit an Analytic View to make changes to the data sources, the hierarchies, or the measures.

To edit an Analytic View, click the **Action** icon on the Analytic View item, then click **Edit Analytic View**. On the Edit Analytic View screen, select a tab and make changes as desired.

When you have completed the changes, click **Update**.

Working with Analyses

Analyses are a collection of multiple reports on a single page, which provides quick access to multiple data analyses collected from different Analytic Views.

Analyses enable you to monitor performance, create reports and set estimates and targets for future work. It provides you a visual representation of performance with charts and graphs.

You can access the Analyses page by clicking the Analyses tile on the Data Analysis home page.

Viewing Analyses

Analyses provide you an insight into the performance of your data.

You can use the Analysis and the Analyze pane to search or browse Analytic Views, view their analysis, or reports you have access to. Clicking on the Analyses takes you to a page where you can view the Analyze pane. Here you can view default hierarchy level and measures selected. You can drag and drop any levels and measures from the Analytic View browser to rows/columns and Values in the drop area respectively. This defines your analysis criteria. Once the values are dropped, the Data Analysis tool generates a query internally. The tool displays the results of the analysis in the form of reports in the Analyses that matches your analysis criteria. You can add multiple reports to the Analysis. You can also examine and analyze the reports and save them as a new analysis. You can just save the Analysis and not a single report. Once you save all the reports, it will be part of that single Analysis. Reports are unnamed.

Workflow to build Analyses

Here is the workflow to build an analyses.

Following are few common tasks to start building Analyses:

- **Create a useful analysis:** Before creating your first analysis, you can construct a useful analysis over a single Analytic View. This way you can generate analyses on which you can create reports that you display on the Analyses.
- **Create Analysis:** Create an analysis to display data from analysis.
- **Create Reports:** An Analysis can have multiple reports that are independent of one another. This can be used to compare and analyze data generated from different Analytic Views.
- **Save Analysis:** Create customized Analyses that enable you to view reports and their analyses in the current state and save it for future reference.

Creating Analyses

Use the Data Analysis tool to create and edit your Analyses. The analysis provides you customized view of Analytic View data. An analysis consists of one or more reports that displays the results of analysis.

You can create a Basepay analysis and add content to track your team's pay. You can view the analysis in a pivot view or tabular view or in the form of charts. You can create an analysis that displays these three views.

In this example, you create a new analysis called New_Analysis.

1. On the Data Analysis home page, click the **Create Analysis** button.
2. You can make changes to existing Analyses by adding different values from hierarchies and measures.
3. The AV name you view on the report represents the AV used to create the report. With a different Analytic View you can create a different report.

> **Note:**
>
> You must have at least one report to build an analysis.

4. To edit the existing analysis, In the Analytic View browser, select the objects to analyze in the navigation pane and drag and drop them to the drop area in Columns, Rows or Values and Filters of the Analyze tab.
5. The report updates based on the artifacts (levels and measures) you select.

The new analysis which contains the updated report will now be visible in the Data Analysis home page for further editing.

- Saving Analyses
  You can save the personalized settings you made for the Analysis and use them on any other Analyses for future reference.

Saving Analyses

You can save the personalized settings you made for the Analysis and use them on any other Analyses for future reference.

You won't have to make these decisions manually every time you open the Analyses page if you save these preferences.

1. Open the analysis for editing from the Data Analysis home page. Select **Edit** from the Analysis tile you wish to edit.
2. Click on the **Save As** icon. Enter a descriptive name for your Analysis.
3. Click **Save**.

Creating Reports

A single report you generate analyzes an AV based on the Levels and measures you select.

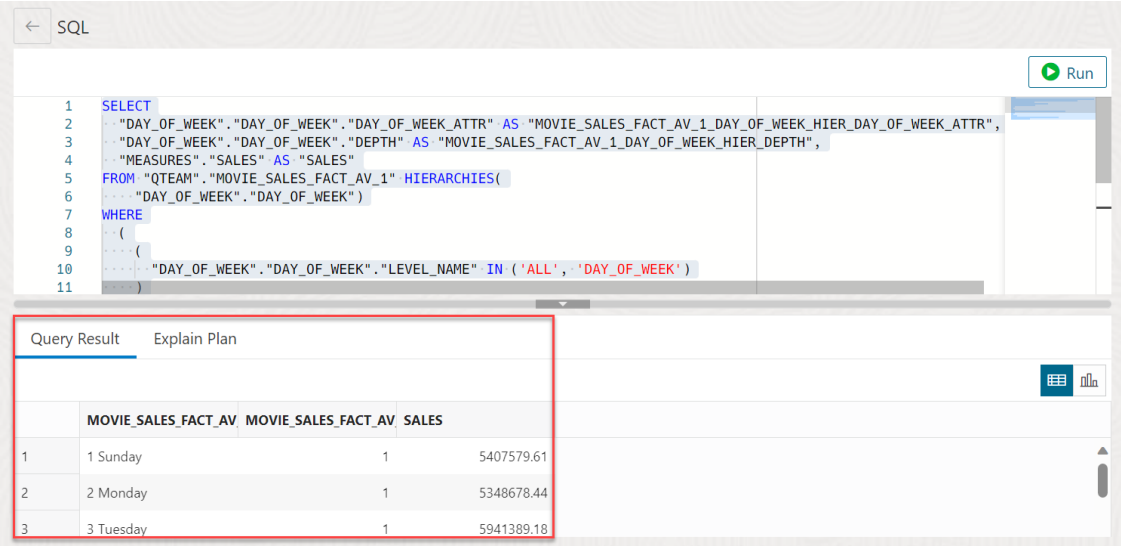
You can add multiple reports to the newly created analysis. One report is independent of another report. To add a report in the Analysis.

1. Open the Analysis for editing from the Data Analysis home page. Select **Edit** from the Analysis tile you wish to edit.
2. Click on the **+ Report** icon to add one or more reports to the Analysis. You can use a report to add configured Analyses to the Analyses page.
3. Click on the report to select it. The resize arrow in the report resizes the report window.
4. Click the cross icon on the selected report to delete the report from the Analysis.
5. The header displays the name of the Analytic View you select.
6. You can expand or collapse the report with their respective arrows.

When you add a report to the Analysis, the report provides the following actions:

- Edit SQL
- Performance
- Rename Report: Click Rename Report to rename it. Click **Save Report** to save the current report.
- Delete Report: Click **Delete Report** to delete the report.

Edit SQL
You can view the SQL output when you click **Edit SQL**. The lower right pane in SQL displays the output of the operation executed in the SQL editor. The following figure shows the output pane on the SQL page.



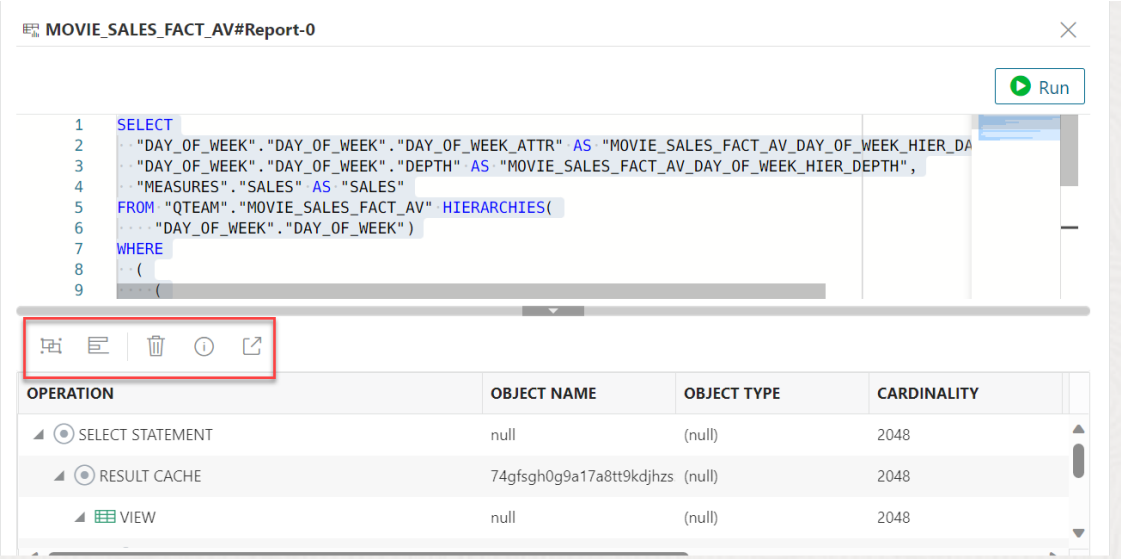
The output pane has the following tabs:

- **Query Result**: Displays the results of the most recent Run Statement operation in a display table.
- **Explain Plan**: Displays the plan for your query using the Explain Plan command. The default view is the diagram view. For more information, see the description of Explain Plan Diagram in subsequent sections.

Performance menu
The Performance menu displays the PL/SQL procedures in the worksheet area which describes the reports associated with the Analytic Views.

The top part of the performance output consists of the worksheet editor for running SQL statements and an output pane to view the results in different forms. You can view the results in a Diagram View, Chart View, Clear Output from the SQL editor, Show info about the SQL statements and Open the performance menu in a new tab.

The following figure shows the output pane of the performance menu:



The output pane has the following tabs:

- **Diagram View**: Displays the plan of your query in the diagram view.
- **Chart View**: Displays the plan of the query in a chart view.
- **Clear Output**: Clears the PL/SLQ statements from the worksheet.
- **Show info**: Displays the SQL statement for which the output is displayed.
- **Open in new tab**: Opens the explain plan in a new window. An Explain Plan displays the plan for your query.

Explain Plan Diagram
The Explain Plain diagram view is a graphical representation of the contents of the insert row statements in the SQL Query. The plan depicts the hierarchical nature of the steps in the execution plan.

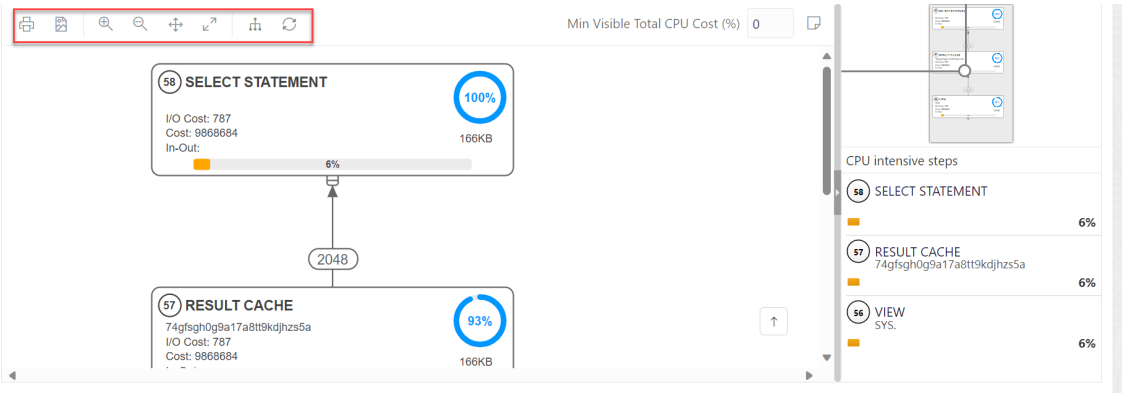
By default, three levels of steps are visible in the diagram. You can use the **+/-** signs at the

bottom of each step (available when the step has children) to expand or collapse. Use Expand All in the toolbar to view all steps in the diagram.
The diagram also provides the following details:

- Cardinality (number on the arrow to the parent step), which is the number of rows processed
- Operation and options applied in that step
- Execution order, which is the sequential number in the order of execution
- Access predicates CPU cost in percentage (orange bar)
- Total CPU cost for the step in percentage (blue circle)
- Estimated I/O Cost, Bytes processed, and Cost metrics

You can see a brief description pop-up when you hover over any of these statistics in a step.

The icons in the toolbar are:



- **Advanced View**: This is the default view of the query when you click Performance. Displays data from SQL Query in mixed tabular/tree view. There is a Diagram View icon that you can use to switch back to the diagram view.
- **Chart View**: Displays data from the SQL query in the form of charts.
- **Print Diagram**: Prints the diagram.
- **Save to SVG**: Saves the diagram to file
- **Zoom In, Zoom Out**: If a step is selected in the diagram, clicking the Zoom In icon ensures that it remains at the center of the screen.
- **Fit Screen**: Fits the entire diagram in the visible area.
- **Actual Size**: Sets the zoom factor to 1.
- **Expand All**: Displays all steps in the diagram.
- **Reset Diagram**: Resets the diagram to the initial status, that is, only three levels of steps are displayed.
- **Show Info**: Shows the SELECT statement used by the Explain Plan functionality.
- **Open in New Tab**: Opens the diagram view in a new tab for better viewing and navigation. The diagram is limited to the initial SELECT statement.
- **Min Visible Total CPU Cost(%)**: Defines the threshold to filter steps with total CPU cost less than the provided value. Enter a value between 0 and 100. There is no filtering for 0.
- **Plan Notes**: Displays the Explain Plan notes.

Properties in Explain Plan Diagram

Double-click or press **Enter** on a selected step to open the Properties slider, which provides more information about that step. See PLAN_TABLE in Oracle Database Reference for a description of each property. The Properties slider shows:

- Displays information for that step extracted from PLAN_TABLE in a tabular format. Nulls are excluded. You can select **JSON** to view the properties in JSON format.
- Information from OTHER_XML column of PLAN_TABLE. The information is displayed in JSON format.

- Working with Reports
  Reports help you in analyzing Analytic Views and Queries.
- Creating Reports on a Query
  This section describes the steps to create reports on an SQL query.

- Creating Reports on an Analytic View
  This section describes the steps to create reports on an Analytic View:

Working with Reports

Reports help you in analyzing Analytic Views and Queries.

The reports are based on the levels and measures you select for the Analytic View and the columns you select for a query.

Click **Analyze** in the Analytic View and click the Table you want to analyze to view the Analyses page.

The Analyses page comprises the following components:

1. **Analytic View Browser**: Select **Analytic View** from the drop-down if you choose to create reports on Analytic Views to view an Analytic View browser. The Analytic View browser displays the Hierarchies, Levels, and Measures associated with the selected Analytic View.

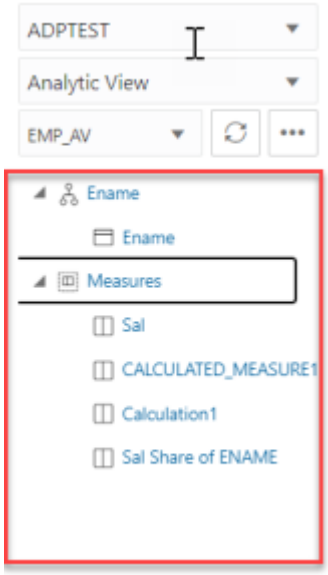
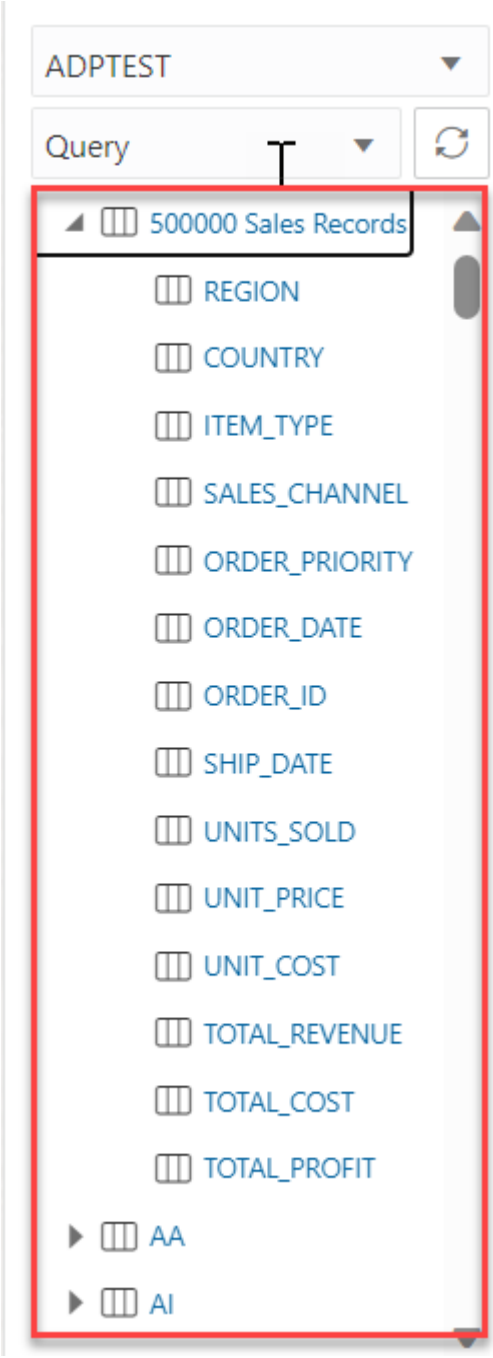
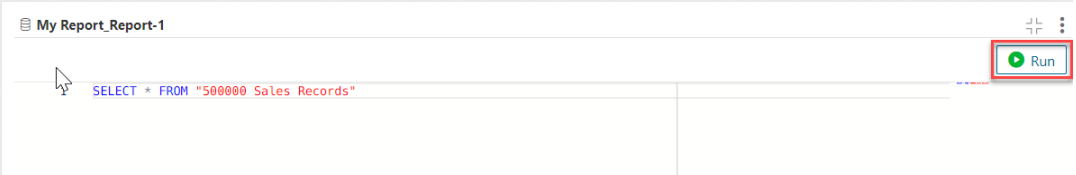


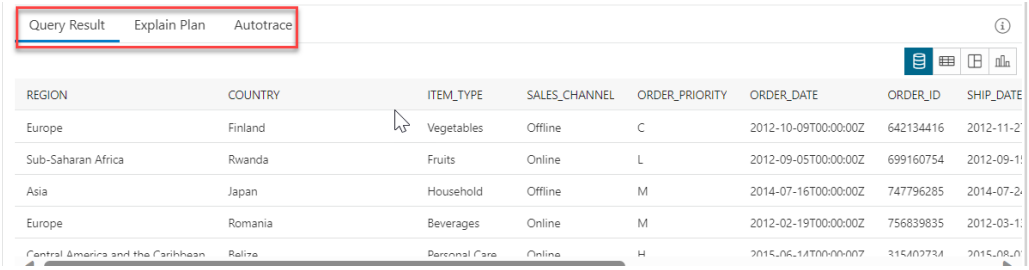
Table Browser: Select **Query** from the drop-down if you choose to create a report on SQL query to view a Table browser. If you select SQL Query, the Table browser displays the available tables and their corresponding columns. You can drill down tables to view their corresponding columns.



2. **SQL Worksheet editor with the Run icon:** You can view this component only when you generate a report on a SQL Query and not an Analytic View.
The SQL editor area enables you to run SQL statements and PL/SQL scripts from the tables you want to query displayed on the Table browser. By default, the SQL editor displays the Select * statement to display all the columns from the first table. Click **Run** to run the statements in the editor.



3. **Output pane:** The output pane, when you view the results of a SQL Query, consists of the following tabs:
   - **Query Result:** Displays the results of the most recent Run Statement operation in a display table.
   - **Explain Plan:** Displays the plan for your query using the Explain Plan command. For more information, refer to the **Explain Plan Diagram** in Creating Reports section.
   - **Autotrace:** Displays the session statistics and execution plan from `v$sql_plan` when running an SQL statement using the Autotrace feature.



4. **Modes of visualization in the Query Result tab**: You can select any of the four modes to visualize the results of the SQL query report you generate.

The four modes of visualization, when you view the reports generated on a SQL query, are:

- **Base Query**: This type of view is by default. Query written in the SQL editor is the Base Query.
- **Table**: You can view the SQL results in tabular form. By selecting this view, a Column drop zone appears which enables you to drag and drop selected columns from the Table browser. By dropping the selected columns in the drop zone, you can view only those columns in the Query Result tab. Select the cross mark beside the Column name to remove it from the drop zone.
- **Pivot**: You can view the results of the SQL query in pivot format. By selecting this format, a Columns, Rows, and Values drop zone appears where you can drag and drop the selected columns, rows or values from the Tables browser.

> **Note:**
>
> Values must be a NUMBER type.

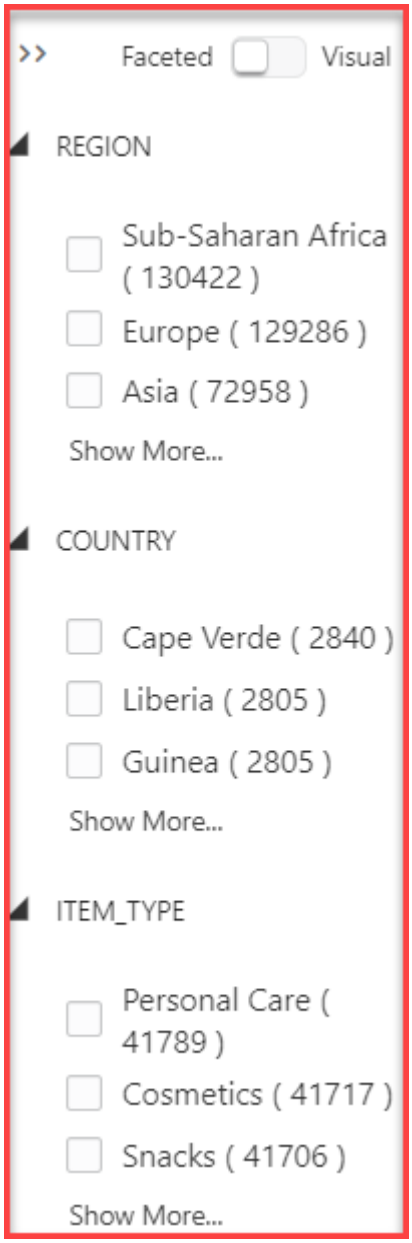
- **Chart**: You can view the SQL results in the form of a chart. By selecting this view an X-axis and Y-axis drop zone appears. Drag and drop selected columns from the Table browser to the drop zone. You must ensure that only the columns with NUMERIC data type can be dropped in the Y axis. Otherwise, the display result would fail with a `Must be a NUMBER type` error. You can add multiple values to the Y-axis. To view the results in the chart view of only a particular y axis, select the Y axis value from the drop-down.

5. **Modes of visualization of reports generated on an Analytic View:** You can select any of the three modes to visualize the results of the report you generate on an Analytic View.

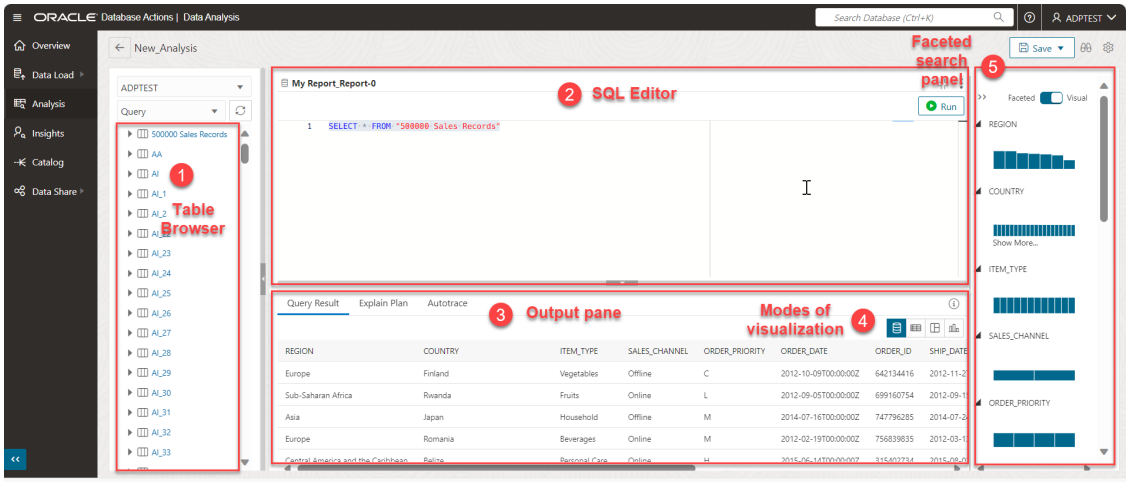
The three modes of visualization when you view the reports generated on an Analytic View are:

- **Table:** You can view the SQL results in tabular form. By selecting this view, a Rows and Filters drop zone appears which enables you to drag and drop selected Hierarchies and Measures from the Analytic View browser. This way you can view the report results that consist of the selected hierarchies and measures.
- **Pivot**: You can view the results of the Analytic View report in the pivot format. By selecting this format, a Columns, Rows,Values and Filters drop zone appears where you can drag and drop the selected hierarchies and measures from the Analytic View browser.Note: Values must be a NUMBER type.
- **Chart**: You can view the report you generate on an Analytic View in the form of a chart. By selecting this view an X-axis, Y-axis, and Filters drop zone appears. Drag and drop selected columns from the Table browser to the drop zone. You must ensure that only the columns with NUMERIC data type can be dropped in the Y axis. Otherwise, the display result would fail with a `Must be a NUMBER type` error. You can add multiple values to the Y-axis. You can select Horizontal and Vertical from the drop-down to view Horizontal and Vertical Charts respectively. You also have the option to select Area Chart, Bar Chart, Line Chart, and Pie Chart from the drop-down.

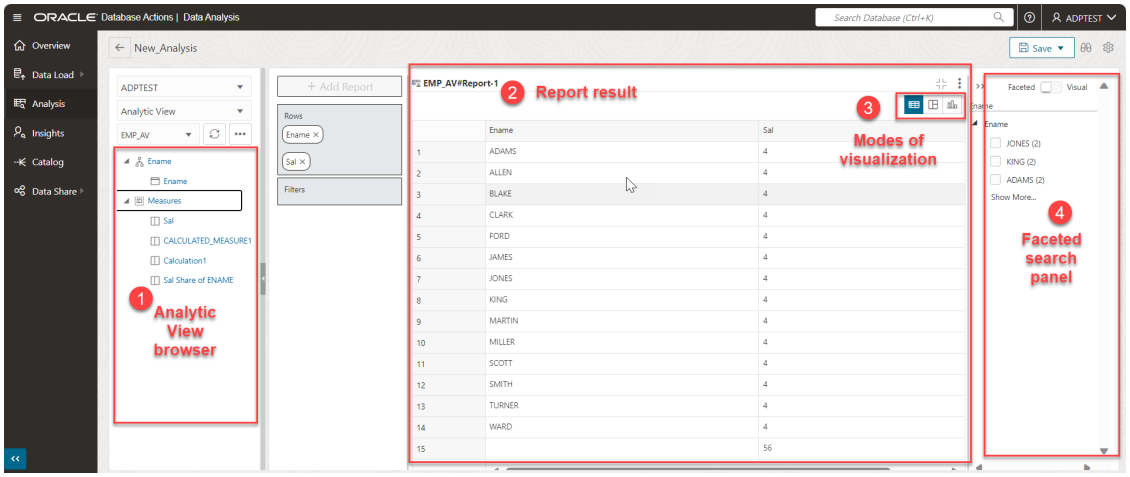
6. **Faceted search panel:** For the reports you generate on a SQL query and an Analytic View, you can view a Faceted search column. For a SQL report, this panel allows you to add filters to the report. The tool generates a filter for each value in the column that is retrieved from the query result. You can filter different columns on the faceted search panel and view the results in the Query result to get only the data you wish to view. You can view the data retrieved from the SQL query in either text or visual format. For reports you generate on an Analytic View, select **Faceted** from the radio button. This filter behaves differently than the Faceted search you generate on an SQL Query. See *Adding filters to a report you generate on an Analytic View*.



The Analyses page when you create a report on an SQL query looks like this:



The Analyses page when you create a report on an Analytic View looks like this:



The following topics describes how to create a report and access the Analyses page:

Creating Reports

1. You can create Reports using either of the following ways:
   - On a Query
   - On an Analytic View
2. Adding filters to a report you generate on an Analytic View

Creating Reports on a Query

This section describes the steps to create reports on an SQL query.

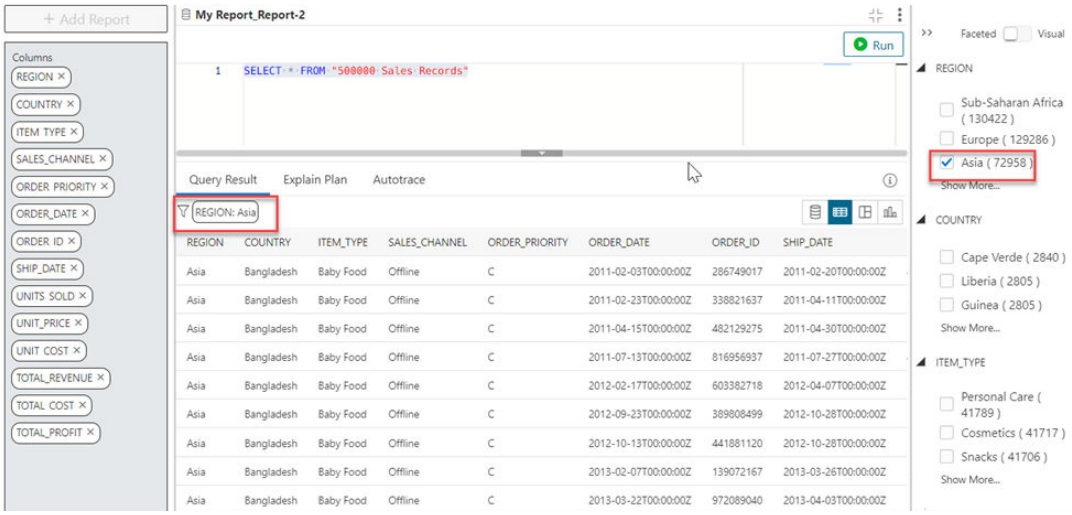
1. From the Analysis home page, select any of the Tables you want to create a report on. You will view the Analyses page with a default query displayed on the SQL editor.

> **Note:**
>
> By default, you will view “Select * from the <Tablename> you select.

2. Click **Run** to run the SQL statement.

   The Query Results tab displays the result in whichever mode you select. The default view is Base Query.

3. Add a filter to the report by displaying the Sales Records of only **Asia** region. Select **Asia** from the faceted search panel.



4. You will view a funnel icon in the Query Result tab which displays a filter with the **REGION** column as **Asia**. The Query Result will display only the records with **REGION** as **Asia**.

Creating Reports on an Analytic View

This section describes the steps to create reports on an Analytic View:

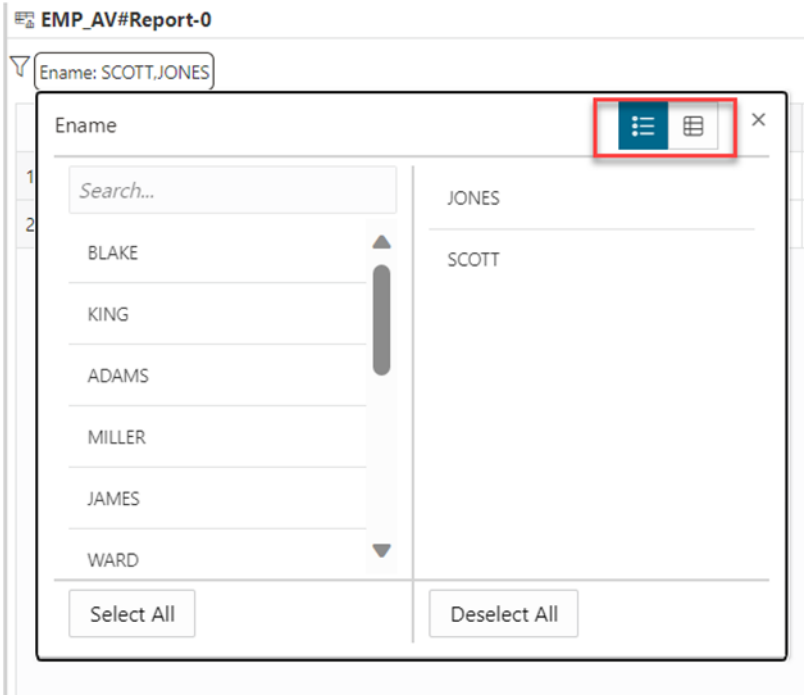
1. From the Analysis home page, under the Analytic View section, select any of the Analytic Views you want to create a report on. You will view the Analyses page with a default report displayed as the output.
2. Drag and drop hierarchies and measures from the Analytic View browser to edit the results you view in the output. For more information, refer to Working with Reports.
3. Click **Expand Report** to expand the view of the report and click **Collapse Report** to minimize the view of the report. The default view of the report you generate on an Analytic View is Pivot.

Adding filters to a report you generate on an Analytic View

Let us add a filter to the report you generated on an Analytic View. Let's say you wish to view the salary of an employee named **SCOTT**.

1. From the Analyses page which displays the report you generated on an **Employee Analytic View**, select “**Scott** from the faceted search panel. You will view the report displaying the result of employees named **Scott**.
2. You can select more filters by selecting the values from the faceted search panel, or by clicking the funnel.

3. Clicking the funnel icon displays all the values of the Employee name column. Select **Jones** to filter the report results further displaying the salary of employees named **Scott** and **Jones**. You can view the values in a list view or a multi select view.



4. Select **Deselect All** to remove all filters.

You can now view the original report result that does not consist of any filters.

Using Calculation Templates

The Data Analysis tool provides templates for all of the calculations typically in demand for business intelligence applications.

The following topics describe the types of calculations available as calculation templates in the tool.

- Cumulative Aggregates
- Prior and Future Period
- Period to Date
- Parallel Period
- Moving Aggregates
- Share
- Rank

- Cumulative Aggregates
  Cumulative calculations start with the first time period and calculate up to the current member, or start with the last time period and calculate back to the current member.
- Prior and Future Period
  The Data Analysis tool provides several calculations for prior or future time periods.

- Period to Date
  Period-to-date functions perform a calculation over time periods with the same parent up to the current period.
- Parallel Period
  Parallel periods are at the same level as the current time period, but have different parents in an earlier period. For example, you may want to compare current sales with sales for the prior year at the quarter and month levels.
- Moving Aggregates
  Moving aggregates are performed over the time periods surrounding the current period.
- Share
  Share calculates the ratio of a measure's value for the current dimension member to the value for a related member of the same dimension.
- Rank
  Rank orders the values of a dimension based on the values of the selected measure. When defining a rank calculation, you choose the dimension, a hierarchy, and the measure.

Cumulative Aggregates

Cumulative calculations start with the first time period and calculate up to the current member, or start with the last time period and calculate back to the current member.

The tool provides several aggregation methods for cumulative calculations:

- **Cumulative Average**: Calculates a running average across time periods.
- **Cumulative Maximum**: Calculates the maximum value across time periods.
- **Cumulative Minimum**: Calculates the minimum value across time periods.
- **Cumulative Total**: Calculates a running total across time periods.

You can choose the measure, the time dimension, and the hierarchy. For selecting the time range see "Choosing a Range of Time Periods" in *Oracle OLAP User's Guide*.

Cumulative Calculation Example

This template defines a calculated measure using Cumulative Minimum.

```
Cumulative minimum of SALES in the TIME dimension and TIME.CALENDAR hierarchy within
ancestor at level TIME.CALENDAR_YEAR. Total from beginning to current member.
```

These are the results of a query against the calculated measure, which displays values for the descendants of calendar year 2021. The minimum value for quarters begins with Q1-21 and ends with Q4-21, and for months begins with Jan-21 and ends with Dec-21.

```
TIME     TIME_LEVEL               SALES  MIN_SALES
-------- -------------------- ---------- ----------
Q1.21    CALENDAR_QUARTER       32977874   32977874
Q2.21    CALENDAR_QUARTER       35797921   32977874
Q3.21    CALENDAR_QUARTER       33526203   32977874
Q4.21    CALENDAR_QUARTER       41988687   32977874
JAN-21   MONTH                  11477898   11477898
FEB-21   MONTH                  10982016   10982016
MAR-21   MONTH                  10517960   10517960
APR-21   MONTH                  11032057   10517960
MAY-21   MONTH                  11432616   10517960
JUN-21   MONTH                  13333248   10517960
JUL-21   MONTH                  12070352   10517960
```

```
AUG-21   MONTH                11108893   10517960
SEP-21   MONTH                10346958   10346958
OCT-21   MONTH                14358605   10346958
NOV-21   MONTH                12757560   10346958
DEC-21   MONTH                14872522   10346958
```

Prior and Future Period

The Data Analysis tool provides several calculations for prior or future time periods.

Here are the calculations used for for prior or future time periods:

- **Prior Period**: Returns the value of a measure at an earlier time period.
- **Difference From Prior Period**: Calculates the difference between values for the current time period and an earlier period.
- **Percent Difference From Prior Period**: Calculates the percent difference between the values for the current time period and an earlier period.
- **Future Period**: Returns the value of a measure at a later time period.
- **Difference From Future Period**: Calculates the difference between the values for the current time period and a later period.
- **Percent Difference From Future Period**: Calculates the percent difference between the values for the current time period and a later period.

When creating a calculation for prior or future time periods, you choose the measure, the time dimension, the hierarchy, and the number of periods from the current period.

Prior Period Example

This template defines a calculated measure using Prior Period:

```
Prior period for measure SALES in TIME dimension and TIME.CALENDAR hierarchy 1 period
ago.
```

These are the results of a query against the calculated measure. The `PRIOR_PERIOD` column shows the value of Sales for the preceding period at the same level in the Calendar hierarchy.

```
TIME     TIME_LEVEL           SALES PRIOR_PERIOD
-------- -------------------- ---------- ------------
2020     CALENDAR_YEAR         136986572    144290686
2021     CALENDAR_YEAR         140138317    136986572
Q1.20    CALENDAR_QUARTER       31381338     41988687
Q2.20    CALENDAR_QUARTER       37642741     31381338
Q3.20    CALENDAR_QUARTER       32617249     37642741
Q4.20    CALENDAR_QUARTER       35345244     32617249
Q1.21    CALENDAR_QUARTER       36154815     35345244
Q2.21    CALENDAR_QUARTER       36815657     36154815
Q3.21    CALENDAR_QUARTER       32318935     36815657
Q4.21    CALENDAR_QUARTER       34848911     32318935
```

Period to Date

Period-to-date functions perform a calculation over time periods with the same parent up to the current period.

These functions calculate period-to-date:

- **Period to Date**: Calculates the values up to the current time period.
- **Period to Date Period Ago**: Calculates the data values up to a prior time period.

- **Difference From Period to Date Period Ago**: Calculates the difference in data values up to the current time period compared to the same calculation up to a prior period.
- **Percent Difference From Period To Date Period Ago**: Calculates the percent difference in data values up to the current time period compared to the same calculation up to a prior period.

When creating a period-to-date calculation, you can choose from these aggregation methods:

- Sum
- Average
- Maximum
- Minimum

You also choose the measure, the time dimension, and the hierarchy.

Period to Date Example

This template defines a calculated measure using Period to Date.

```
Gregorian Year to date for SALES in the TIME dimension and TIME.CALENDAR hierarchy.
Aggregate using MINIMUM from the beginning of the period.
```

These are the results of a query against the calculated measure. The `MIN_TO_DATE` column displays the current minimum `SALES` value within the current level and year.

```
TIME     TIME_LEVEL               SALES MIN_TO_DATE
-------- ------------------- ---------- -----------
Q1.21    CALENDAR_QUARTER       36154815    36154815
Q2.21    CALENDAR_QUARTER       36815657    36154815
Q3.21    CALENDAR_QUARTER       32318935    32318935
Q4.21    CALENDAR_QUARTER       34848911    32318935
JAN-21   MONTH                   13119235    13119235
FEB-21   MONTH                   11441738    11441738
MAR-21   MONTH                   11593842    11441738
APR-21   MONTH                   11356940    11356940
MAY-21   MONTH                   13820218    11356940
JUN-21   MONTH                   11638499    11356940
JUL-21   MONTH                    9417316     9417316
AUG-21   MONTH                   11596052     9417316
SEP-21   MONTH                   11305567     9417316
OCT-21   MONTH                   11780401     9417316
NOV-21   MONTH                   10653184     9417316
DEC-21   MONTH                   12415325     9417316
```

Parallel Period

Parallel periods are at the same level as the current time period, but have different parents in an earlier period. For example, you may want to compare current sales with sales for the prior year at the quarter and month levels.

The Data Analysis tool provides several functions for parallel periods:

- **Parallel Period**: Calculates the value of the parallel period.
- **Difference From Parallel Period**: Calculates the difference in values between the current period and the parallel period.
- **Percent Difference From Parallel Period**: Calculates the percent difference in values between the current period and the parallel period.

To identify the parallel period, you specify a level and the number of periods before the current period. You can also decide what happens when two periods do not exactly match, such as comparing daily sales for February (28 days) with January (31 days).

You also choose the measure, the time dimension, and the hierarchy.

Parallel Period Example

This template defines a calculated measure using Parallel Period.

```
Parallel period for SALES in the TIME dimension and TIME.CALENDAR hierarchy 1
TIME.CALENDAR.QUARTER ago based on position from beginning to ending of period.
```

These are the results of a query against the calculated measure, which lists the months for two calendar quarters. The parallel month has the same position within the previous quarter. The prior period for `JUL-21` is `APR-21`, for `AUG-21` is `MAY-21`, and for `SEP-21` is `JUN-21`.

```
TIME    PARENT     SALES   LAST_QTR
------- ---------- ---------- ----------
APR-21   CY2006.Q2    11356940   13119235
MAY-21   CY2006.Q2    13820218   11441738
JUN-21   CY2006.Q2    11638499   11593842
JUL-21   CY2006.Q3     9417316   11356940
AUG-21   CY2006.Q3    11596052   13820218
SEP-21   CY2006.Q3    11305567   11638499
```

Moving Aggregates

Moving aggregates are performed over the time periods surrounding the current period.

The Data Analysis tool provides several aggregation methods for moving calculations:

- **Moving Average**: Calculates the average value for a measure over a fixed number of time periods.
- **Moving Maximum**: Calculates the maximum value for a measure over a fixed number of time periods.
- **Moving Minimum**: Calculates the minimum value for a measure over a fixed number of time periods.
- **Moving Total**: Returns the total value for a measure over a fixed number of time periods.

You can choose the measure, the time dimension, and the hierarchy. You can also select the range, as described in "Choosing a range of time periods" in *Oracle OLAP User's Guide* , and the number of time periods before and after the current period to include in the calculation.

Moving Aggregates Example

This template defines a calculated measure using Moving Minimum.

```
Moving minimum of SALES in the TIME dimension and TIME.CALENDAR hierarchy. Include 1
preceding and 1 following members within level.
```

These are the results of a query against the calculated measure, which displays values for the descendants of calendar year 2021. Each value of Minimum Sales is the smallest among the current value and the values immediately before and after it. The calculation is performed over all members of a level in the cube.

```
TIME     TIME_LEVEL           SALES  MIN_SALES
-------- -------------------- ---------- ----------
Q1.21    CALENDAR_QUARTER       32977874   32977874
Q2.21    CALENDAR_QUARTER       35797921   32977874
```

```
Q3.21     CALENDAR_QUARTER        33526203   33526203
Q4.21     CALENDAR_QUARTER        41988687   31381338
JAN-21    MONTH                   11477898   10982016
FEB-21    MONTH                   10982016   10517960
MAR-21    MONTH                   10517960   10517960
APR-21    MONTH                   11032057   10517960
MAY-21    MONTH                   11432616   11032057
JUN-21    MONTH                   13333248   11432616
JUL-21    MONTH                   12070352   11108893
AUG-21    MONTH                   11108893   10346958
SEP-21    MONTH                   10346958   10346958
OCT-21    MONTH                   14358605   10346958
NOV-21    MONTH                   12757560   12757560
DEC-21    MONTH                   14872522   12093518
```

Share

Share calculates the ratio of a measure's value for the current dimension member to the value for a related member of the same dimension.

You can choose whether the related member is:

- **Top of hierarchy**: Calculates the ratio of each member to the total.
- **Member's parent**: Calculates the ratio of each member to its parent.
- **Member's ancestor at level**: Calculates the ratio of each member to its ancestor, that is, a member at a specified level higher in the hierarchy.

When creating a share calculation, you can choose the measure, dimension, and hierarchy. You also have the option of multiplying the results by 100 to get percentages instead of fractions.

Share Example

This template defines a calculated measure using `SHARE`:

```
Share of measure SALES in PRODUCT.PRIMARY hierarchy of the PRODUCT dimension as a ratio
of top of hierarchy.
```

These are the results of a query against the calculated measure. The `TOTAL_SHARE` column displays the percent share of the total for the selected products.

```
PRODUCT            PROD_LEVEL         SALES TOTAL_SHARE
-------------------- --------------- ---------- -----------
Total Product        TOTAL            144290686         100
Hardware             CLASS            130145388          90
Desktop PCs          FAMILY            78770152          55
Portable PCs         FAMILY            19066575          13
CD/DVD               FAMILY            16559860          11
Software/Other       CLASS             14145298          10
Accessories          FAMILY             6475353           4
Operating Systems    FAMILY             5738775           4
Memory               FAMILY             5430466           4
Modems/Fax           FAMILY             5844185           4
Monitors             FAMILY             4474150           3
Documentation        FAMILY             1931170           1
```

Rank

Rank orders the values of a dimension based on the values of the selected measure. When defining a rank calculation, you choose the dimension, a hierarchy, and the measure.

You can choose a method for handling identical values:

- **Rank**: Assigns the same rank to identical values, so there may be fewer ranks than there are members. For example, it may return `1, 2, 3, 3, 4` for a series of five dimension members.
- **Dense Rank**: Assigns the same minimum rank to identical values. For example, it may return `1, 2, 3, 3, 5` for a series of five dimension members.
- **Average Rank**: Assigns the same average rank to identical values. For example, it may return `1, 2, 3.5, 3.5, 5` for a series of five dimension members.

You can also choose the group in which the dimension members are ranked:

- **Member's level**: Ranks members at the same level.
- **Member's parent**: Ranks members with the same parent.
- **Member's ancestor at level**: Ranks members with the same ancestor at a specified level higher in the hierarchy.

Rank Example

This template defines a calculated measure using Rank:

```
Rank members of the PRODUCT dimension and PRODUCT.PRIMARY hierarchy based on measure
SALES. Calculate rank using RANK method with member's parent in order lowest to highest.
Rank NA (null) values nulls last.
```

These are the results of a query against the calculated measure in which the products are ordered by `RANK`:

```
PRODUCT             SALES       RANK
------------------- ---------- ----------
Monitors               4474150          1
Memory                 5430466          2
Modems/Fax             5844185          3
CD/DVD                16559860          4
Portable PCs          19066575          5
Desktop PCs           78770152          6
```

Oracle Autonomous Database Add-in for Excel

The Oracle Autonomous Database Add-in for Excel integrates Microsoft Excel spreadsheets with the Autonomous Database to retrieve and analyze data from Analytic Views in the database. You can also directly run SQL queries to view their results in the worksheet.

- Install the add-in on Mac
  The Oracle Autonomous Database add-in for Excel is supported on Mac OS running Microsoft Office 365.
- Install the add-in on Windows
  The Oracle Autonomous Database Add-in for Excel is supported on Windows 10 and Windows 11 operating systems running Microsoft Excel 365.
- Uninstall the add-in
  The following section describes the steps to uninstall the Oracle Autonomous Database add-in.
- Using Oracle Autonomous Database Add-in for Excel
  After you install the add-in, a new ribbon tab, **Autonomous Database** appears in MS Excel.

- Connection management
  Each time you start the add-in for Excel, you must create a connection.
- Import a Connection
  You can import a connection file that you can download from the Database Actions launchpad. This file is in JSON format.
- Add a connection
  You can manually create a connection to an autonomous database. Adding a connection allows you to specify the connection credentials to the database in which you will connect to the schema of your Autonomous Database.
- Share a connection
  You can import or export a connection using the Import and Export buttons on the Connections panel.
- Run native SQL queries in an Excel worksheet
  The Oracle Autonomous Database add-in for Excel lets you run native SQL queries to work with your data in an Excel worksheet.
- Query an Analytic View in an Excel worksheet
  The Query Wizard menu enables you to query an Analytic View and retrieve the results in an Excel Worksheet. Once the wizard retrieves the data, it becomes local to Excel. You can further edit the data in Excel but not write back to the Autonomous Database.
- Data Analysis in Excel Sheet
  The Data Analysis panel enables you to perform SQL queries.
- FAQs for Troubleshooting errors with Excel Add-in
  If you experience issues with Oracle Autonomous Database Add-in for Excel, refer to frequently asked questions in this section to identify and resolve issues.

Install the add-in on Mac

The Oracle Autonomous Database add-in for Excel is supported on Mac OS running Microsoft Office 365.

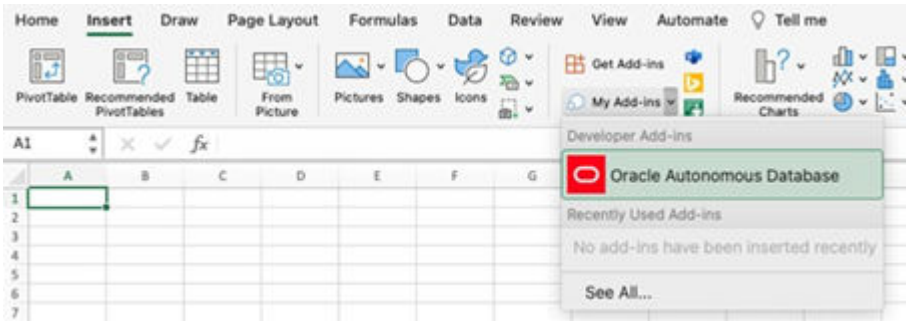
To install the Autonomous Database Add-in for Excel, run the installer script file from your Autonomous Database instance by following the steps below:

- Open the Database Actions Launchpad.
- On the **Downloads** tab of the Database Actions page, click the **Download Microsoft Excel/Google Sheets Add-in** pane.

- Click **Download**.
- Click the Microsoft Excel tab and select the **Download Add-in** button to download the **oracleplugin.zip** file.
- You can now view the zip file in the Downloads folder.
- Create a new folder named **Add-in** on your system.
- Extract the contents of the zip file in the **Add-in** folder.

Follow these steps to install the add-in.

1. Quit Excel before you run the installer.
2. Navigate to the *install.sh* file in the **Add-in** folder.
3. Right-click *install.sh* and select the following options as shown : **Open With** -> **Other…** -> **Enable: All Applications** ->**Utilities**->**Terminal**
4. On completion, close the Terminal window.
5. Start Excel and open a new or existing workbook.
6. From the **Insert** menu in the Excel ribbon, select the drop-down menu of **My Add-ins**. A new **Oracle Autonomous Database** entry appears under the **Developer Add-Ins** dialog box.



7. Select **Oracle Autonomous Database**.

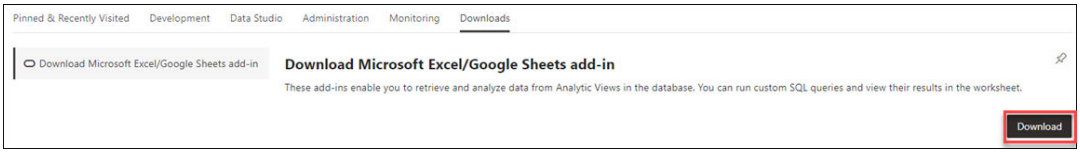
A new **Autonomous Database** ribbon tab appears in MS Excel.

Install the add-in on Windows

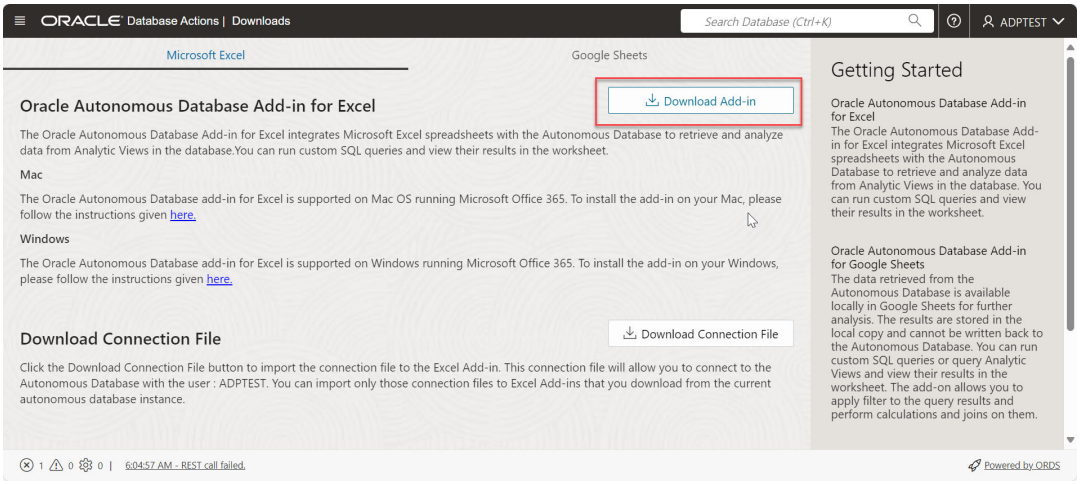
The Oracle Autonomous Database Add-in for Excel is supported on Windows 10 and Windows 11 operating systems running Microsoft Excel 365.

To install the Autonomous Database Add-in for Excel, download the *oracleplugin.zip* file and extract it to get the *install.cmd* script file from your Database Actions instance.

- Open the Database Actions Launchpad.
- On the **Downloads** tab of the Database Actions page, click the **Download Microsoft Excel/Google Sheets Add-in** pane.



- Click **Download**.
- Click the **Download Add-in** icon in the Microsoft Excel tab to download the Oracle Autonomous Database Add-in for Excel.



- Extract the *oracleplugin.zip* file to a new folder in the Downloads of your system. The extracted folder consists of an installer (install.cmd file), a manifest.xml file and a readme.txt file.

Follow these steps to install the add-in.

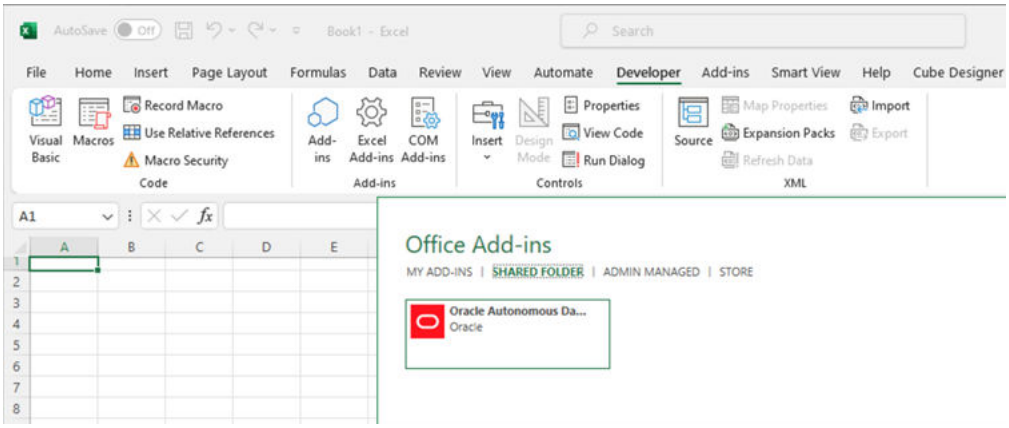
1. Quit Excel before you run the installer.
2. Right-click the *install.cmd* file that you downloaded.

> **Note:**
>
> After running the installer on Windows, the add-in automatically creates a network share and adds the shared location as a trusted catalog location for Office add-ins. A catalog is used to store the manifest for the Excel Add-in. It enables publishing and management of the Excel add-in as well as other add-ins that are available in the Office Store and licensed for corporate use. You can acquire the Excel add-in by specifying the shared manifest folder as a trusted catalog.

3. Select **Run as administrator**.

> **Note:**
>
> You must have Administrator privileges to install the Excel add-in for Oracle Autonomous Database successfully.

4. Start Excel and open a new or existing workbook.
5. From the **Developer** menu in the Excel ribbon, click **Add-ins**, select the **SHARED FOLDER** tab on the pop-up window and select **Oracle Autonomous Database**.



6. After you install the add-in, a new **Autonomous Database** ribbon tab appears in MS Excel.

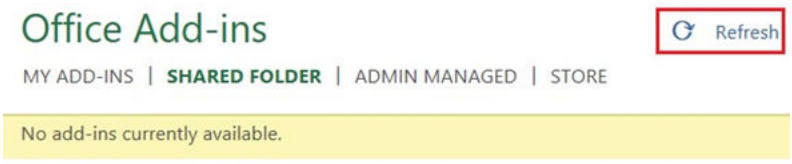
> **Note:**
>
> You can re-run the installer after the initial installation. Re-run the installer and choose the option of your preference. You can repair your existing installation by deleting it, selecting the installed trusted catalog or adding another manifest to the working installation.

Uninstall the add-in

The following section describes the steps to uninstall the Oracle Autonomous Database add-in.

To uninstall the Oracle Autonomous Database Add-in for Excel for Windows:

- Delete the *manifest.xml* file from the folder located in *%LOCALAPPDATA%\Oracle\Autonomous Database\manifest* .
- Click **Refresh** in the Office Add-ins window to remove the Autonomous Database tab from MS Excel.



No add-ins will now be available from the Shared folder of the Office Add-ins window.

> **Note:**
>
> After uninstalling the Add-in, if you re-install it from a different Autonomous Database (ADB), the add-in attempts to load the old ADB. You must check if the shared manifest folder's location (share path) points to the correct location. For more details, refer to *Configuring the Excel Trusted Add-in Catalog* in FAQs for Troubleshooting errors with Excel Add-in.

To uninstall the Oracle Autonomous Database Add-in for Excel for Mac:

- Enter the following command in the terminal to remove the *manifest.xml* file.

```
rm ~/Library/Containers/com.microsoft.Excel/Data/Documents/wef/manifest.xml
```

The Oracle Autonomous Database Add-in is uninstalled from Excel for Mac.

Using Oracle Autonomous Database Add-in for Excel

After you install the add-in, a new ribbon tab, **Autonomous Database** appears in MS Excel.

You can connect to multiple Autonomous Databases, work with Analytic Views, tables, and Views, and view the data in the worksheet.

This ribbon provides buttons that let you connect to the Autonomous Database.

Click **Connections** to connect to an Autonomous Database. You must Refer to the *Connection management* for more details.

Click **Settings** to view the logging level settings of the Excel Add-in. You can also clear the logs or export the log files by copying the logging information to the clipboard.

Click **About** to view the Add-in and the supported Excel versions. The About window also displays whether the spreadsheet is connected to the database. It also shows version information for the database and Oracle Rest Data Services.

Click **Native SQL** to run custom SQL queries.

Click **Query Wizard** to select the Analytic View you want to query. You can review and edit the query, add or edit filters and calculations, and choose the output format from tabular and pivot formats.

Click **Analyses and Reports** to view the Analyses and reports the Excel Add-in created using the web UI.

Click **Data Analysis** to query an existing Analytic View and run queries.

Selecting **Native SQL** icon or **Query Wizard** icon from the ribbon launches the Oracle Autonomous Database wizard in the Excel task pane.

Connection management

Each time you start the add-in for Excel, you must create a connection.

The connections feature lets you manage and connect to multiple Autonomous Databases with a single add-in. Multiple connections can be created. However, only one connection can remain active.

The connection panel lets you connect to the Autonomous Database through a connection where you provide the login credentials and access the Autonomous Database.

Within the Connections panel, you can:

- Create or delete multiple connections using a single add-in.
- Share connection information by exporting and importing connection information to a file.
- View existing connections.
- Refresh connections to retrieve updated data from the Autonomous Database and view the connection status of the Excel Add-in.

Selecting **Connections** opens the Connections wizard.

> **Note:**
>
> This is an implicit type of connection. Refer to Authenticate using Implicit connection to understand more about implicit connection.

The Connections wizard consists of the following four buttons at the header:

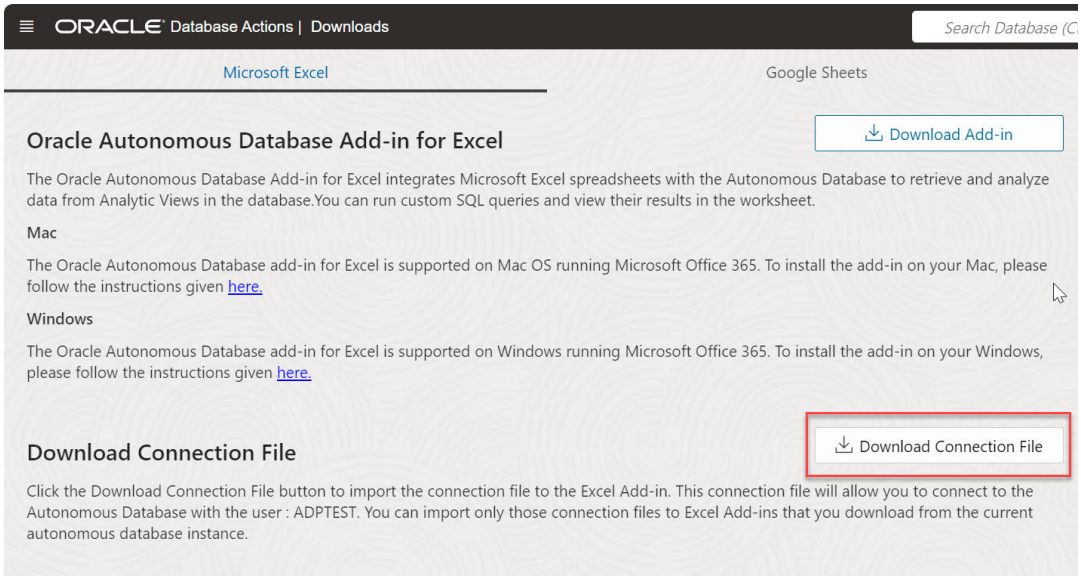
- **Refresh:** You can refresh the connection with this icon. The green icon besides the connection name indicates that the connection is active. A red icon beside the connection name suggests you are not connected to the database.
- **Add:** Select **Add** to Add a Connection. Refer to *Add a Connection* section for more details.
- **Export:** Select **Export** to export connections. Refer to the *Share a Connection* section for more details.
- **Import:** Select **Import** to launch the import wizard to choose a connection file. These files are in JSON format.

Import a Connection

You can import a connection file that you can download from the Database Actions launchpad. This file is in JSON format.

Follow the below steps from the Database Actions instance, to download the connection file.

1. From the Downloads page of the Database Actions instance, click the **Download Connection File** icon to download the connection file in your system.

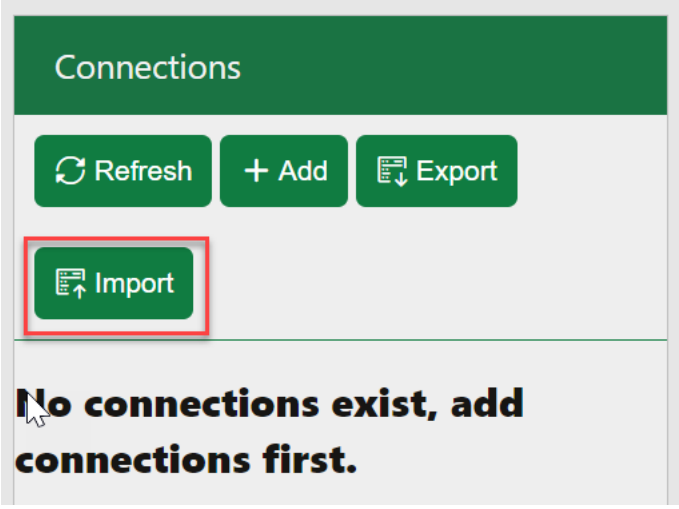


2. The file is downloaded onto your system.

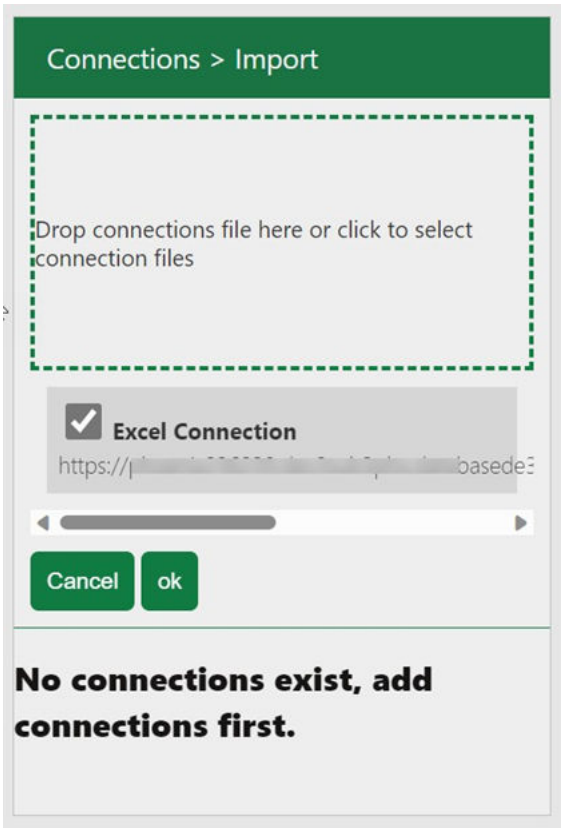
> **Note:**
>
> This connections file can be used with the add-in installed from the same Autonomous Database instance.

To import a connection:

1. Select **Connections** from the Autonomous Database menu in the Excel sheet. This opens the Connections wizard.
2. Click **Import** to import the connection file you downloaded from the Database Actions instance.



3. Click and drop the connection file from your system to the drop area of the wizard. After the connection file loads, select the check box beside the connection file you want to import from the file.



> **Note:**
>
> A connection file can have multiple connections in it. You can import a connection you downloaded from the same Autonomous Database instance. If you use the add-in to connect to a different Autonomous Database, you must manually create a connection. Refer to the Add a Connection section for more details.

4. Click **ok** to proceed.
5. Click the three vertical dots beside the connection file and select **Connect**.

> **Note:**
>
> If you view a red icon beside the connection file, the connection is inactive or incorrect. Click the three vertical dots beside the connection file and select **Edit** to update the connection file. Ensure the Autonomous Database URL is correct and click **Save**. An example of a correct URL is “https://<hostname>-<databasename>.adb.<region>.oraclecloudapps.com/"

6. Specify the schema name in the username field and the corresponding password in the credentials screen you view. Click **Sign in** to log in to the autonomous database.

You will view “Active Connection” beside the connection name.

Add a connection

You can manually create a connection to an autonomous database. Adding a connection allows you to specify the connection credentials to the database in which you will connect to the schema of your Autonomous Database.

This connection will allow you to use the database from Excel.

1. Click on the **Add** button on the header of the Connections pane to add a connection. This opens an Add new connection dialog box.
2. Specify the following fields on the Add new connection dialog box:
   - **Alias:** Enter the Alias name for the Autonomous Database URL. For readability purposes, Oracle recommends using a name different name from the URL.
   - **Autonomous Database URL:** Enter the URL of the Autonomous Database you wish to connect to. Copy the entire URL from the web UI of the Autonomous Database. For example, enter or copy the following link "https://<hostname>-<databasename>.adb.<region>.oraclecloudapps.com/" to connect to the database. This will be provided to you by the administrator.
   - **Schema:** Enter the schema you use for this connection.
   - **Client ID:** Enter the Client ID for this connection. Refer to the *Generate the client ID for a connection* section to generate the client ID of this implicit connection and paste it on this field.

   Click the **Copy implicit connection query template** button to get the PL/SQL code to generate a client ID required in the Client ID field. Refer to the *Generate the client ID for a connection* section below for details.
3. Click **Save** to save the connection.

You should be able to view the new connection now.

- Generate Client ID for a connection
  The OAuth Client key is generated using SQL.

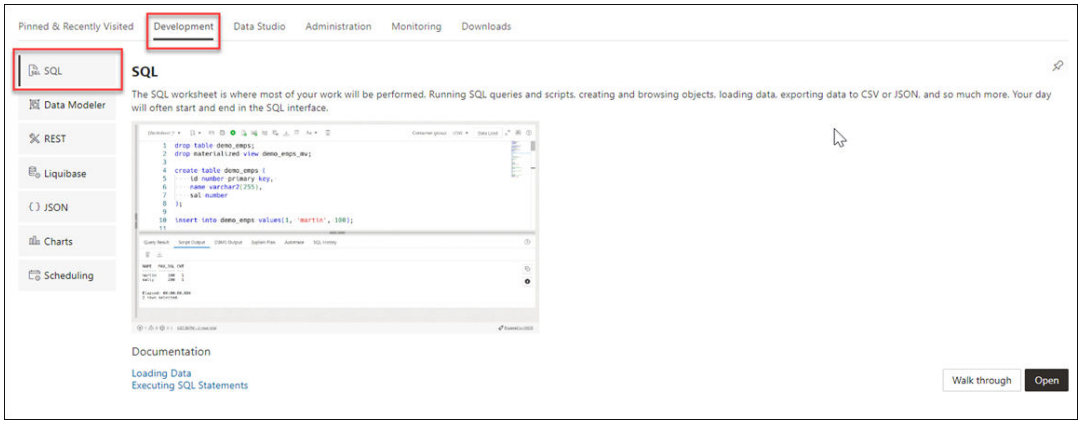
Generate Client ID for a connection

The OAuth Client key is generated using SQL.

The Copy implicit connection query template button copies the connection query template. The template contains PL/SQL code that generates an OAuth Client ID. To create the Client ID, copy and run this PL/SQL code in the worksheet editor.

This section describes how to generate a client ID.

1. On the Development section of the Database Actions Launchpad, select **SQL** card. This opens the SQL page.



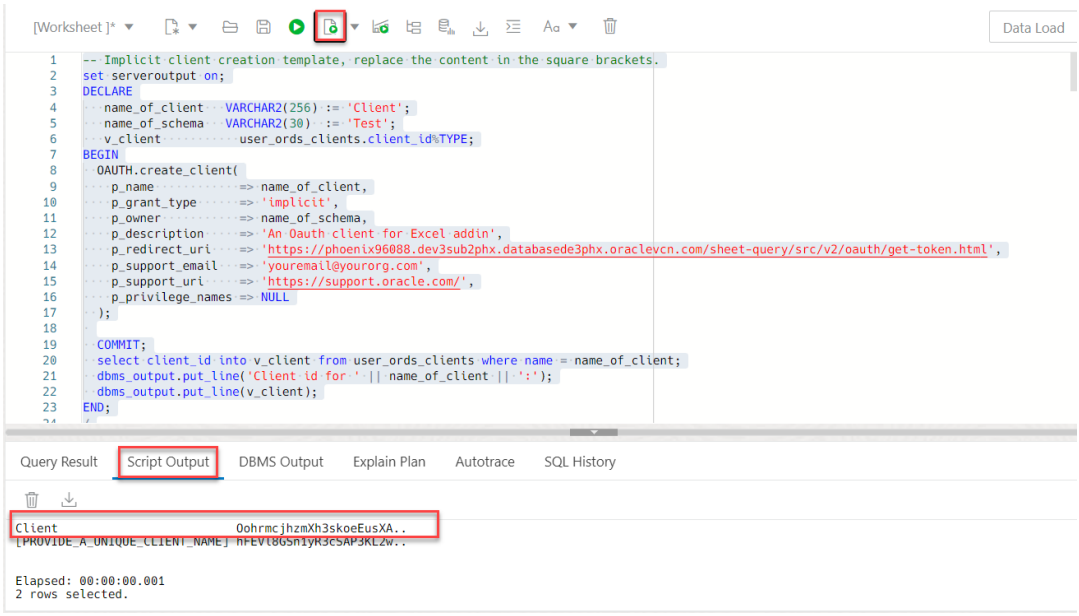
2. Paste the implicit connection query template you copied as explained in the previous section. Here is a sample of the implicit connection query template in the image below.

```
-- Implicit client creation template, replace the content in the square brackets.
set serveroutput on;
DECLARE
   name_of_client   VARCHAR2(256) := '[PROVIDE_A_UNIQUE_CLIENT_NAME]';
   name_of_schema   VARCHAR2(30)  := '[PROVIDE_THE_SCHEMA_NAME]';
   v_client           user_ords_clients.client_id%TYPE;
BEGIN
  OAUTH.create_client(
    p_name            => name_of_client,
    p_grant_type      => 'implicit',
    p_owner           => name_of_schema,
    p_description     => 'An Oauth client for Excel addin',
    p_redirect_uri    => 'https://abcd.dev3sub2phx.databasede3phx.oraclevcn.com/sheet-query/src/v2/oauth/get-token.html',
    p_support_email   => 'youremail@yourorg.com',
    p_support_uri     => 'https://support.oracle.com/',
    p_privilege_names => NULL
  );

  COMMIT;
  select client_id into v_client from user_ords_clients where name = name_of_client;
  dbms_output.put_line('Client id for ' || name_of_client || ':');
  dbms_output.put_line(v_client);
END;
/
-- To list all the clients, run this query:
select name, client_id from user_ords_clients;
```

3. On the worksheet editor, replace the "[PROVIDE_A_UNIQUE_CLIENT_NAME]" text in the variable name field with the client name of your choice. The name has to be unique. For example, no other OAuth client can have the same name as the name you provide in this field.
4. In the worksheet editor, replace the"[PROVIDE_THE_SCHEMA_NAME]" text with your schema name in the variable name field.
5. You could replace the support URI in the Create Client PL/SQL procedure with the email you used to create the OAuth Client. For example, “youremail@yourorg.com".

> **Note:**
>
> Do not change the template otherwise, you might view errors that will cause the unsuccessful creation of the Client ID. The p_redirect_uri field is auto-generated and is different for each Autonomous Database.

6. Click the **Run Script** icon on the worksheet toolbar to run the PL/SQL code.



7. The following is the sample output you will view in the Script Output tab after you run the PL/SQL code.

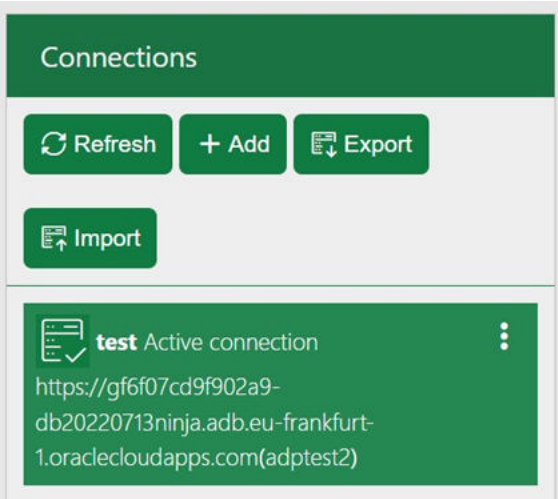
```
PL/SQL procedure successfully completed.

Elapsed: 00:00:00.113

----------------------------------------

NAME                                CLIENT_ID
----------------------------------------------------------
Client                              OohrmcjhzmXh3skoeEusXA..
[PROVIDE_A_UNIQUE_CLIENT_NAME] hFEVl8GSn1yR3cSAP3KL2w..

Elapsed: 00:00:00.170
2 rows selected.
```

8. Copy the client ID from the first line of the script output. You can also copy the client ID equivalent to the client name you provided on the implicit connection query template. Here is the client ID, in the above example, OohrmcjhzmXh3skoeEusXA...
9. Paste the Client ID on the Client ID value field of the Add new connection dialog box. Refer to the Add a connection for more details on this.

Once you have created a new connection, you can view the connection you have added in the Connections panel. A connection in the panel lists the following:

- An alias at the top. For example, **test** is the alias.
- The bottom part of the connection panel displays the URL of the Autonomous Database with the schema you connect to.

- A connection status indicator where the indicator identifies if the connection is connected or not. A red cross mark indicates no connection. Whereas a check mark indicates the connection is successful.
- An actions icon rightmost to the connection panel.



Click the **Actions** icon on the connection. You can perform the following actions on the selected connection:

- **Connect:** Click **Connect** to connect the add-in with the Autonomous Database. This opens the login page of Oracle Database Actions, the Autonomous Database you wish to connect to.
  Enter the schema name in the username field and the corresponding password.

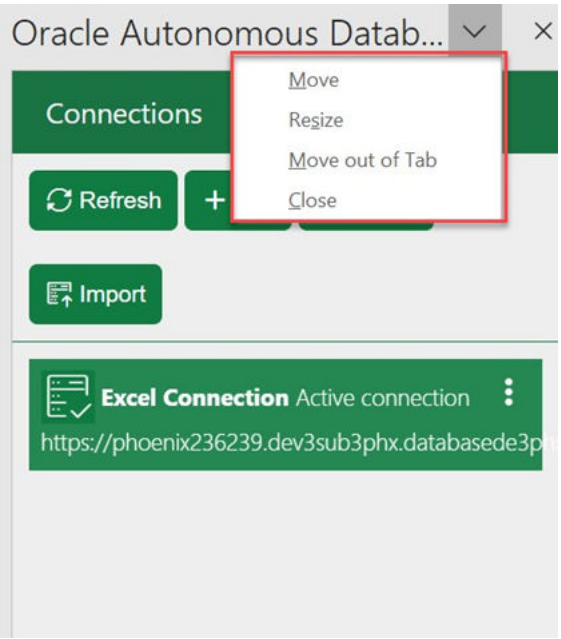
  > **Note:**
  >
  > – The add-in for Excel asks for your permission the first time you log into the database. Select **Approve** to proceed with the login.

  You will view a notification page that says the authorization of the Excel with Autonomous Database is successful.
- **Activate:**
  There can be only one active connection when you connect to multiple Autonomous Databases. Click **Activate** to make the selected connection functional. The Active connection is displayed at the top of the panel. You can expand or collapse the active connection. Expand the active connection to view its detailed status such as the Autonomous database URL, schema and connection status.

  You can collapse in case of spacing issues. You can view the alias and the status of the connection in its collapsed form.
- **Edit:** This button enables you to edit the existing connection. Click **Edit** to review, view or edit connection-based information. Selecting Edit opens the same dialog you view when you add a connection. Edit any information details, such as, Alias, Autonomous Database URL, Schema or the Client ID of the existing connection.

- **Duplicate:** Select **Duplicate** to clone the connection from the list of connections displayed in the Connections panel. This creates a copy of the connection without having to enter the details again.
- **Disconnect:** Select **Disconnect** to disconnect from the connection. Once the connection disconnects, you will see a red cross mark beside the connection name. This indicates that the connection is terminated.
- **Remove:** Select **Remove** to remove the connection from the list of connections displayed in the Connections panel.

Managing the Excel Add-in Panel



Click **Move** in the drop-down of the wizard pane to move the wizard to your preferred location.

The **Resize** option in the drop-down resizes the query window. This option allows you to resize the wizard window by moving the double-headed arrow sideways. The wizard expands when you move the arrow to the left and contracts when you drag it to the right.

Click **Move out of Tab** to move the add-in from the task pane.

Click **Close** to close the wizard.

Share a connection

You can import or export a connection using the Import and Export buttons on the Connections panel.

- **Import:** Click **Import** and select a connection file from your local device. Once you import the connection file, you can view the connection in the Connection panel with a check box beside it. Select the check box and click **OK** to add the connection to the list of connections in the panel. The add-in copies the connection information you can use as a new connection. With the import feature, you do not have to enter the connection information to add a new connection.
- **Export:** The export button exports an existing connection which you can import later. Clicking **Export** opens a check box beside each connection in the connections list. You

can select the connection you wish to export. Multiple selections are allowed. After you select the connection, click **OK**. Once the connection file is exported, you can view that the add-in for Excel downloads the connection file (*.json file) to your local device. The exported connection file is named spreadsheet_addin_connections.json.

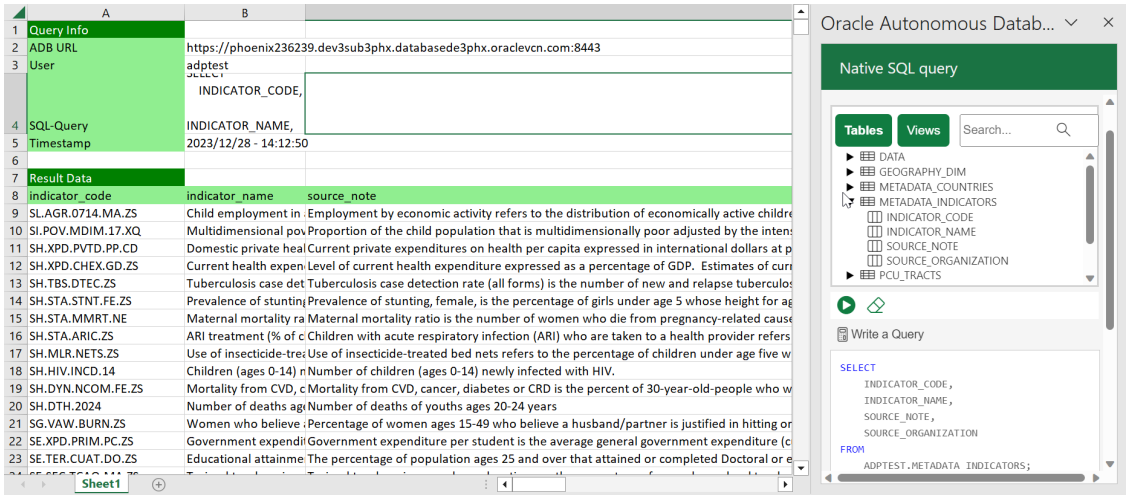
Run native SQL queries in an Excel worksheet

The Oracle Autonomous Database add-in for Excel lets you run native SQL queries to work with your data in an Excel worksheet.

With the add-in, you can create a table and insert, update and delete rows from the existing tables or views. You can view the results in the current worksheet or different worksheets.

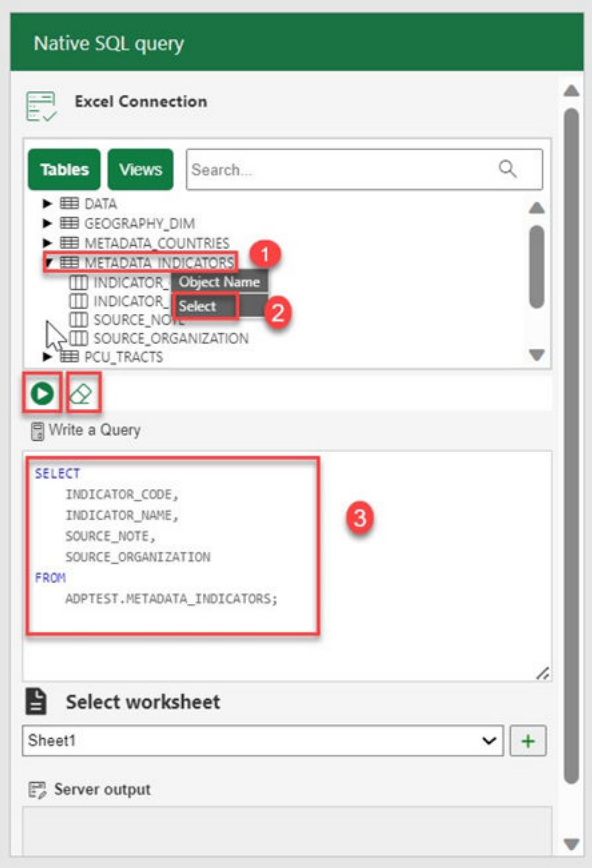
The following image shows your data retrieved from the Autonomous Database and displayed in the worksheet. The Query Info section comprising the Timestamp, User name and SQL Query are shown in Excel. You can edit custom queries and run them. The worksheet displays the results of queries from the retrieved data in tabular format.

The add-in maintains a live connection with the database. However, the data retrieved is local to Excel. In case of inactivity, the connection times out, and you must log in again. You can change the active connection from the connections panel. The image shows the results from a single query, but you can insert many queries in a single workbook.



To run a query using the add-in, run Excel and create a blank workbook using the standard Excel workbook file format.

1. In the Excel ribbon, select the **Autonomous Database**.
2. Click the **Native SQL** icon from the ribbon. This opens an **Oracle Autonomous Database** dialog box in the Excel Task Pane with **Tables** and **Views** icons and a search field beside them.



3. Select **Table** to view all the existing tables in the schema. Click **Views** to see the current views in the schema.
4. You can right-click the table whose data you want to query and choose **Select** to view all the table's columns. The column names will be displayed in the Write a Query section. You can click on the table and view individual columns as well. Click the **Run** button to run the SQL query in the query editor. The query results will be displayed in the worksheet you select.

> **Note:**
>
> You will view an error message if you click the **Run** icon while the query editor is empty.

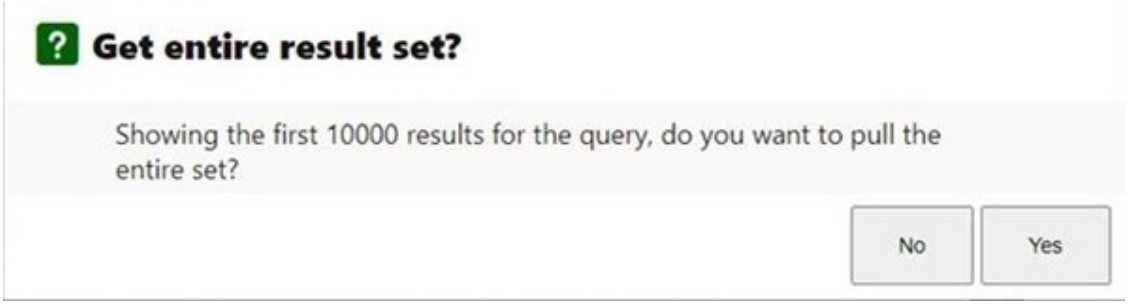
5. You can click **+** sign beside the **Select worksheet** drop-down to display the results in a new worksheet.
6. The worksheet also displays the timestamp, the user who creates and runs the query and the autonomous database URL.

To run another query follow these steps:

1. Click the eraser icon to clear the previous query from the SQL editor and write a new query.
2. In the **Select worksheet** drop-down, select a new sheet, *Sheet 2*, in this case. The Add-in adds a sheet for the user. If you work on the same sheet, the Add-in refreshes the data in the existing worksheet.

3. Click the **Run** icon to display the query results.

The worksheet displays the result of the query at a go. While this behavior works for most scenarios, sometimes, for large data sets, the query result might exceed 10K rows. Although you can view the 10K rows, a confirmation window asks if you want to view the rest of the results.



Select **Yes** to view the entire result set. Loading all the data may take a while. You must fetch all data before working with Pivot tables, or it will lead to incorrect results from aggregation in Pivot tables.

Close the Query Wizard panel to cancel the operation of fetching the result.

> **Note:**
>
> Close the Query Wizard panel to cancel the operation of fetching the result.

Query an Analytic View in an Excel worksheet

The Query Wizard menu enables you to query an Analytic View and retrieve the results in an Excel Worksheet. Once the wizard retrieves the data, it becomes local to Excel. You can further edit the data in Excel but not write back to the Autonomous Database.

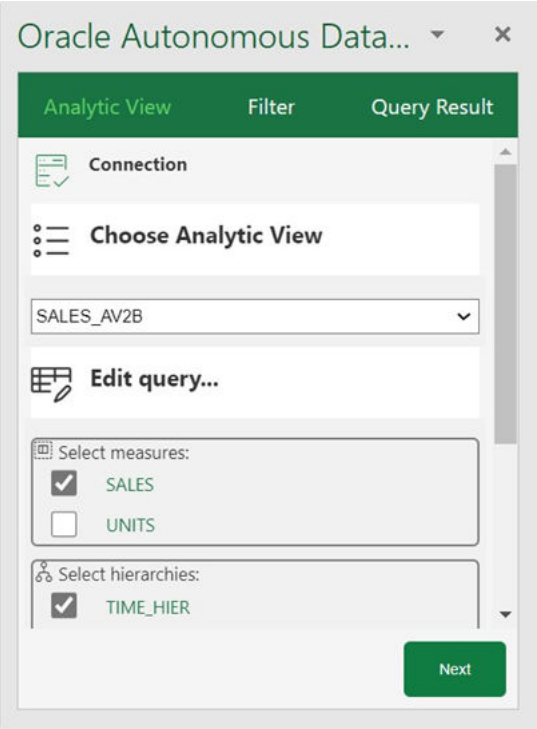
You can query an Analytic View to visualize the result data in the worksheet. You can search for the Analytic View, and select measures, hierarchies, and levels from the query. You can also add filters and calculated measures to the query and view the query result in the spreadsheet.

By default, the data is retrieved in tabular format. You also have the option to create an Excel pivot from this data.

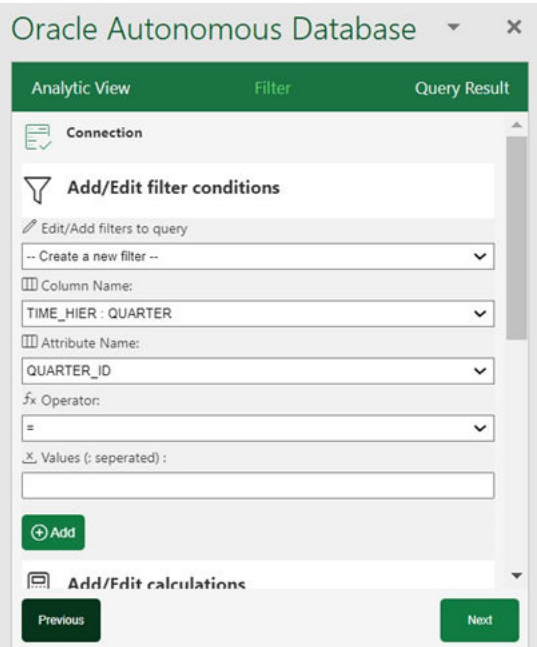
The Query Wizard has three panels:

1. **Analytic View panel:**The Analytic View panel contains a list of Analytic Views from which you build queries. You edit the query by selecting
   - measures
   - hierarchies,
   - and levels

   and progress to the next panel.

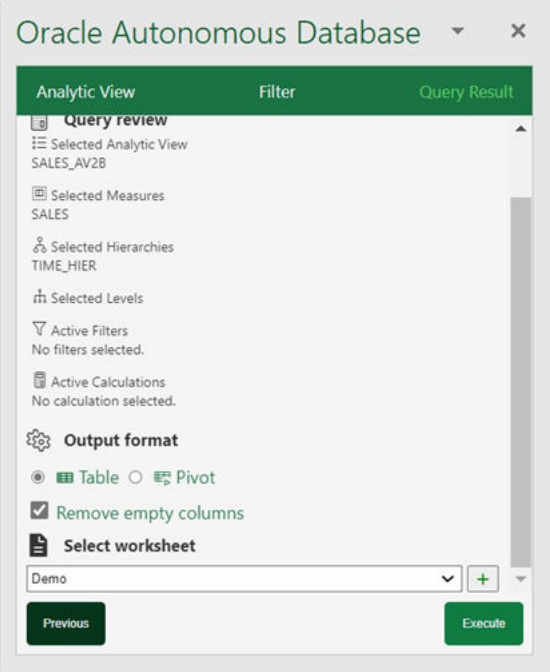


2. **Filter panel:** The Filter panel displays to the right of the Analytic View panel when you click **Next** on the wizard. You can create filter conditions to filter the data and also add manual calculations to the Analytic View query in this panel.



3. **Query Result panel:**When you click **Next** on the wizard, the Query Result panel displays to the right of the Filter panel. You run the query once you select the filter criteria and determine what calculated measures to add to your query. You can view and revise the SQL query. After the SQL query runs, you view the query results in the worksheet. You can

select the output format of the result here. You can view the results in tabular format or a Pivot table.



To query an analytic view and explore the Query Wizard menu in the MS Excel ribbon:

1. On the ribbon, select the Query Wizard icon.
2. Selecting the Query Wizard opens an **Oracle Autonomous Database** dialog box in the Excel Task Pane.
3. Select an existing Analytic View from the drop-down in the Analytic View pane. As you select the Analytic View, it appears on the Analytic View field.
4. Select your choice of measures, hierarchies, and levels the available measures, hierarchies and levels associated with the Analytic View. Click **Next**.
5. The wizard window progresses to the **Filter** pane where you can add or edit filters to query.
6. Under Add or Edit filter conditions, do the following.
   - Select the column name and the attribute name from the drop-down- the values of the attribute change dynamically with the change in column names.
   - Select an operator in the Operator field to apply to the values that you specify in the Value field.
   - Specify a value or values from the list containing your selected column members. You need to enter the value into the Values field manually. For example, you can select > in the Operator field to use only values greater than the value that you select in the Value list. If you select 100,000 from the Value list, the filter uses values from the column greater than 100,000. You can use this information in an analysis to focus on products performing well. For multiple values use ":" as the separator.
   - Click **Add Filter** to add another filter condition.
7. Under Add or Edit Calculations, do the following.
   - Specify the column whose values you want to include in the group or calculated item.

- On the Calc expression field, enter a custom calculated expression you want to perform on the column value. You can add functions or conditional expressions.

8. Click **Next** to progress to the **Query Result**.
9. You can view, edit, and review the query you have generated from the Query Review editor.
10. Select **Remove empty columns** to remove columns with no values returned in the result.
11. Select **Column per level** to retrieve all hierarchy levels in a single column.
12. Select the worksheet from the drop-down where you want to view the result.
13. Click **Execute** to run the query.
14. You can view the result of the query in the worksheet you select.
15. You can always modify the query in the Oracle Autonomous Database dialog box editor even after results are generated.
16. Select **Table** in the Query Result pane to view the results in the worksheet in a tabular format.
17. Select **Pivot** in the Query Result pane to view the results in a new worksheet in Pivot format.

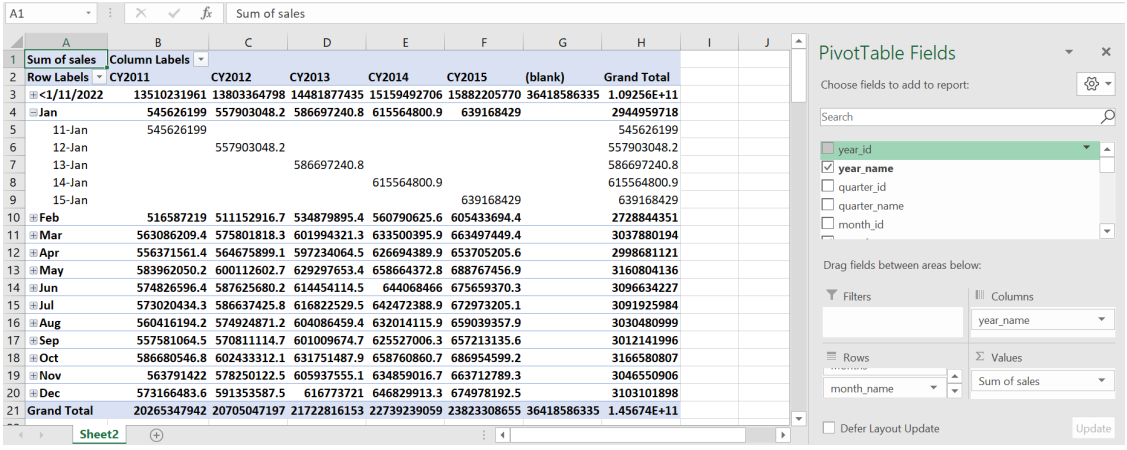
View the results in Pivot tables

A Pivot table view is interactive and allows you to transpose rows and columns. A pivot table can summarize, sort, reorganize, count the total and perform an average of the result data. They are navigable and drillable.

Apart from tabular mode, to view the query results in pivot table mode, select the **Pivot Table** option in the Autonomous Database wizard. Click **Execute** to view the query results in the Pivot table.

Clicking **Execute** opens the query results in a new sheet with a PivotTable Fields wizard. Click anywhere outside the table in the spreadsheet to switch the Pivot Table wizard to the Native SQL query wizard. Select any cell in the table to continue editing the Pivot Table fields.

You can view the **Grand Total** of the entire pivot table in the last row of the table.



Sum of sales	Column Labels						
Row Labels	CY2011	CY2012	CY2013	CY2014	CY2015	(blank)	Grand Total
<1/11/2022	13510231961	13803364798	14481877435	15159492706	15882205770	36418586335	1.09256E+11
Jan	545626199	557903048.2	586697240.8	615564800.9	639168429		2944959718
11-Jan	545626199						545626199
12-Jan		557903048.2					557903048.2
13-Jan			586697240.8				586697240.8
14-Jan				615564800.9			615564800.9
15-Jan					639168429		639168429
Feb	516587219	511152916.7	534879895.4	560790625.6	605433694.4		2728844351
Mar	563086209.4	575801818.3	601994321.3	633500395.9	663497449.4		3037880194
Apr	556371561.4	564675899.1	597234064.5	626694389.9	653705205.6		2998681121
May	583962050.2	600112602.7	629297653.4	658664372.8	688767456.9		3160804136
Jun	574826596.4	587625680.2	614454114.5	644068466	675659370.3		3096634227
Jul	573020434.3	586637425.8	616822529.5	642472388.9	672973205.1		3091925984
Aug	560416194.2	574924871.2	604086459.4	632014115.9	659039357.9		3030480999
Sep	557581064.5	570811114.7	601009674.7	625527006.3	657213135.6		3012141996
Oct	586680546.8	602433312.1	631751487.9	658760860.7	686954599.2		3166580807
Nov	563791422	578250122.5	605937555.1	634859016.7	663712789.3		3046550906
Dec	573166483.6	591353587.5	616773721	646829913.3	674978192.5		3103101898
Grand Total	20265347942	20705047197	21722816153	22739239059	23823308655	36418586335	1.45674E+11

Data Analysis in Excel Sheet

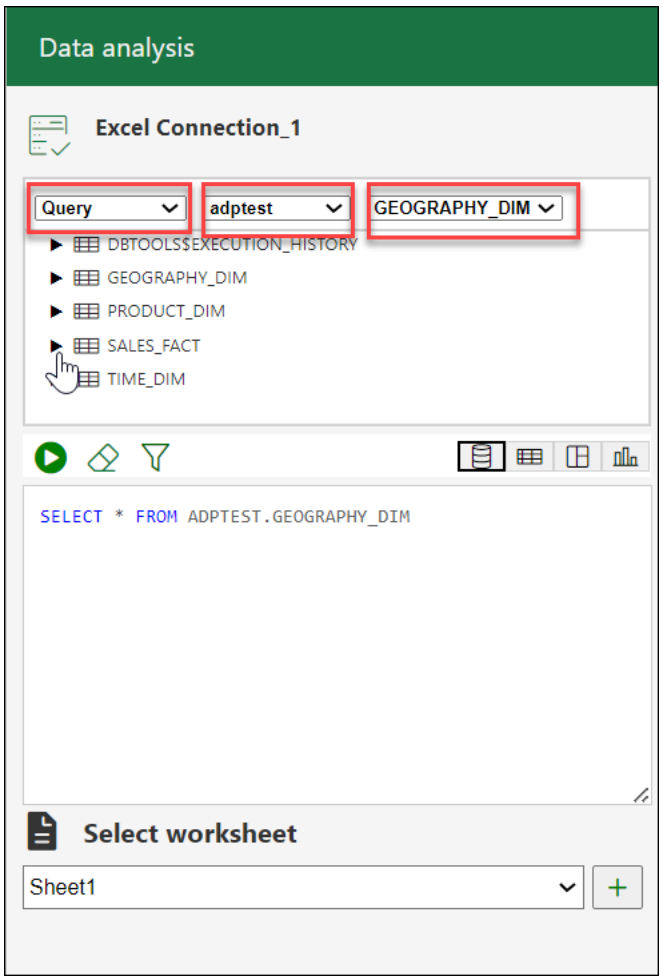
The Data Analysis panel enables you to perform SQL queries.

The add-on enables you to receive a copy of data from the Autonomous Database to the Excel sheet. You can query an existing Analytic View and run SQL Query using the Oracle Autonomous Database wizard. You can retrieve the Analytic View and manipulate the query according to your requirements to visualize the result data in the worksheet. You can search for the Analytic View and select measures, hierarchies, and levels from the query. You can also add filters and calculated measures to the query and view the result in the sheet. By default, the data is retrieved in tabular format.

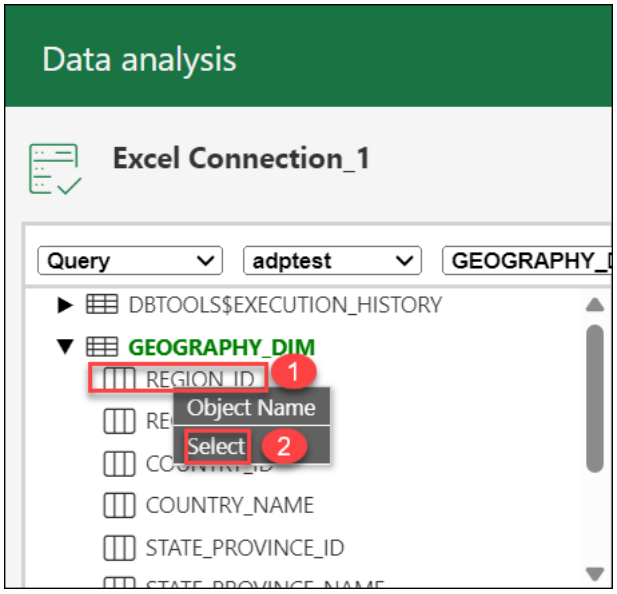
You can run custom queries. The add-on enables you to apply a filter to the query results. The add-in lets you view query results that can be customized with selected columns using a faceted filter.

To run a custom query using the add-in:

1. On the Excel Sheet, select the menu item **Autonomous Database**.
2. Select **Data Analysis**. Selecting Data Analysis opens a Data Analysis wizard. On the Data Analysis wizard, select **Query** from the drop-down, the **schema** you wish to use from the drop-down, and the **table** on which you perform the query.



3. You can select a column of the table and right-click the column, click **Select** to assist the add-in in forming a select query of the column from the table. Alternatively, you can drag and drop the selected column to the query area which enables the wizard to produce a select query of the column in the query display area.

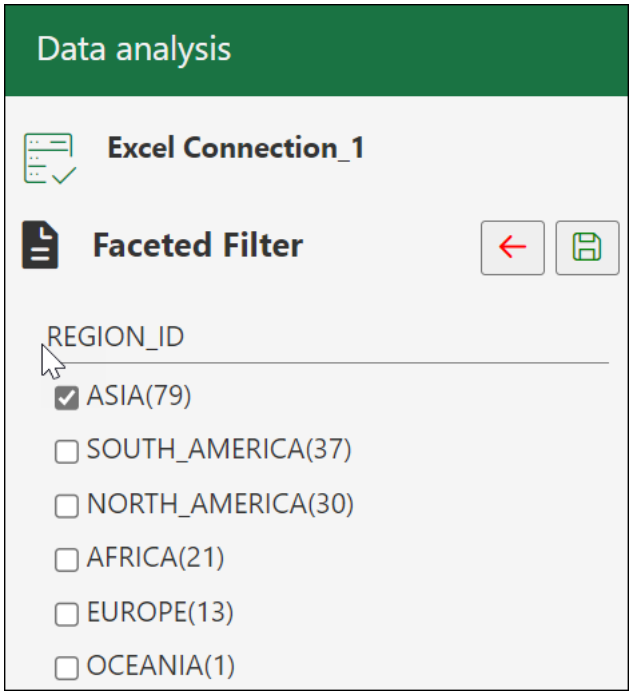


You will view the default query in the query editor area.

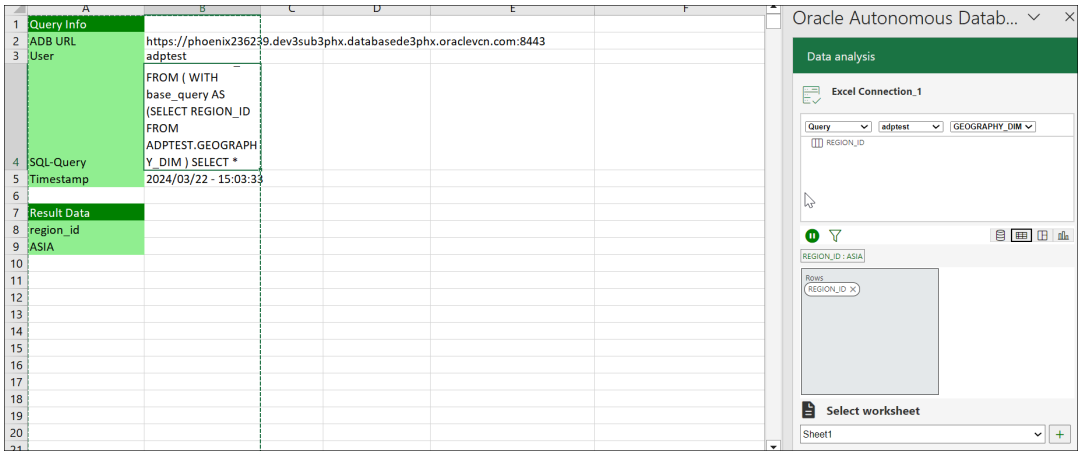
4. You can select any of the four modes to visualize the results of the SQL query report you generate. You can select any of the four modes to visualize the results of the SQL query report you generate:

- **Base Query**: This type of view is by default. The query written in the SQL editor is the Base Query.
- **Table**: You can view the SQL results in tabular form. By selecting this view, a column drop zone appears, enabling you to drag and drop selected columns from the Table browser. Moving the selected columns in the drop zone allows you to view only those columns in the Result data generated in the worksheet. Select the cross mark beside the Column name to remove it from the drop zone.
- **Pivot**: You can view the SQL query results in pivot format. By selecting this format, an X and Y drop zone appears where you can drag and drop the selected columns from the Tables browser to the drop area.
- **Chart**: You can view *Area Chart*, *Bar Chart*, *Line Chart*, or *Pie Chart* when you select this option.. The mappings displayed when you select one of the options are as follows:
  - **Orientation**: Choose between horizontal and vertical orientation types from the drop-down list.
  - **X axis label and Y axis label**: Optionally enter labels for X axis and Y axis.

5. Click the **funnel icon** (Faceted Filter) to add filters to the result. The wizard generates a filter for each value in the column retrieved from the query result. You can filter different columns on the faceted filter panel and view the results in the worksheet to view only the data you wish to view. For example, to view the customer reports by Region, click the **faceted filter** and select **Asia** under `Region_ID`.



6. Click **Save** to view the results. Click **Back** to go back to the main wizard.
7. Select **Run** to generate the results of the custom query in the worksheet. Click **Pause** to make any changes to the query, such as updating the columns of the table without updating the worksheet.

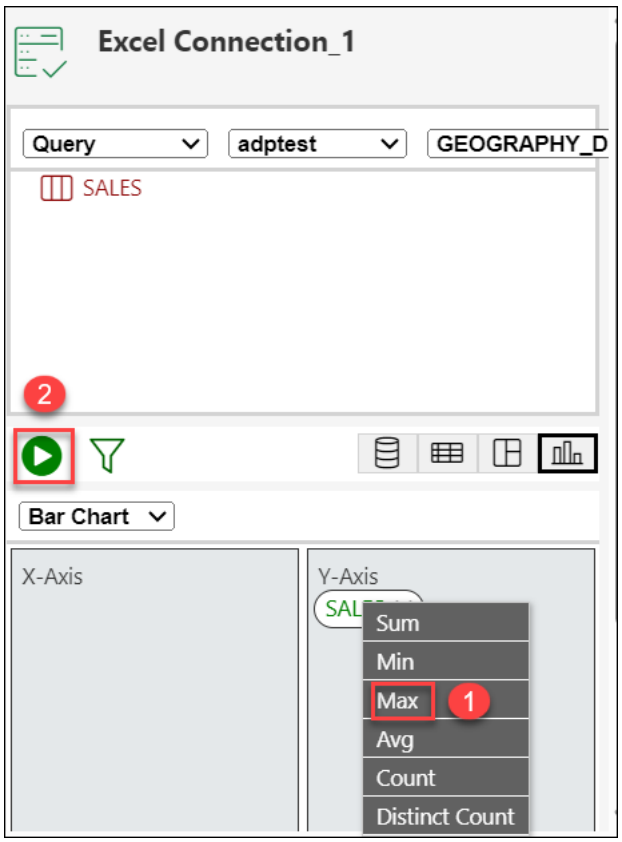


Perform aggregate functions using the Excel add-in

You can also perform aggregate functions such as `SUM`, `MIN`, `MAX`, `AVG`, `COUNT`, and `DISTINCT COUNT`. In this example, we're primarily going to focus on using the Data Analysis feature to gain insights from our sales data.

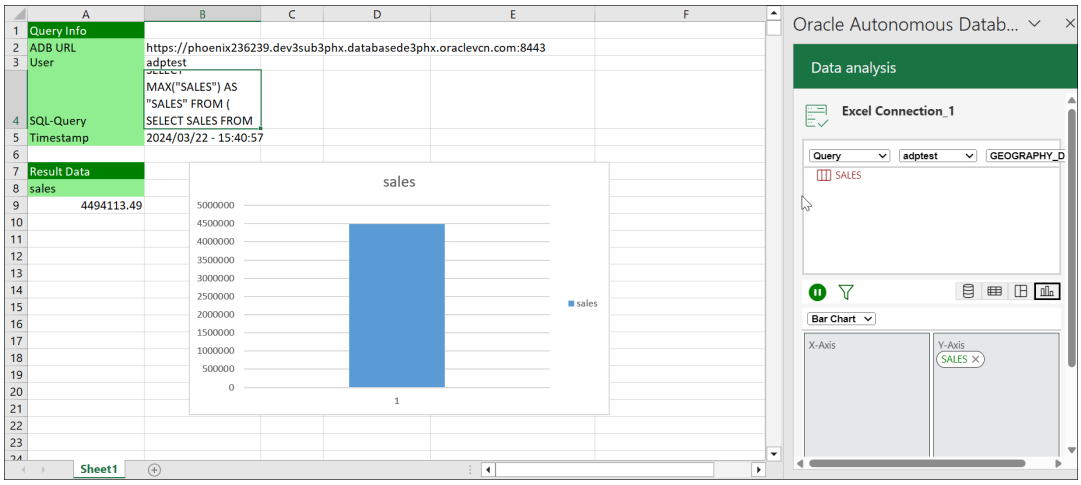
1. Select **Data Analysis**. Selecting Data Analysis opens a Data Analysis wizard. On the Data Analysis wizard, select **Query** from the drop-down, the **schema** you wish to use from the drop-down, and the **table** on which you perform the query.
   Drag and drop the sales value to the query editor and click **Chart** to view the sales in chart format.

2. To calculate the maximum sale value, right-click the sales value and select **Max** from the list of available aggregate functions.



Click **Run** to generate the maximum sales amount in the chart format.

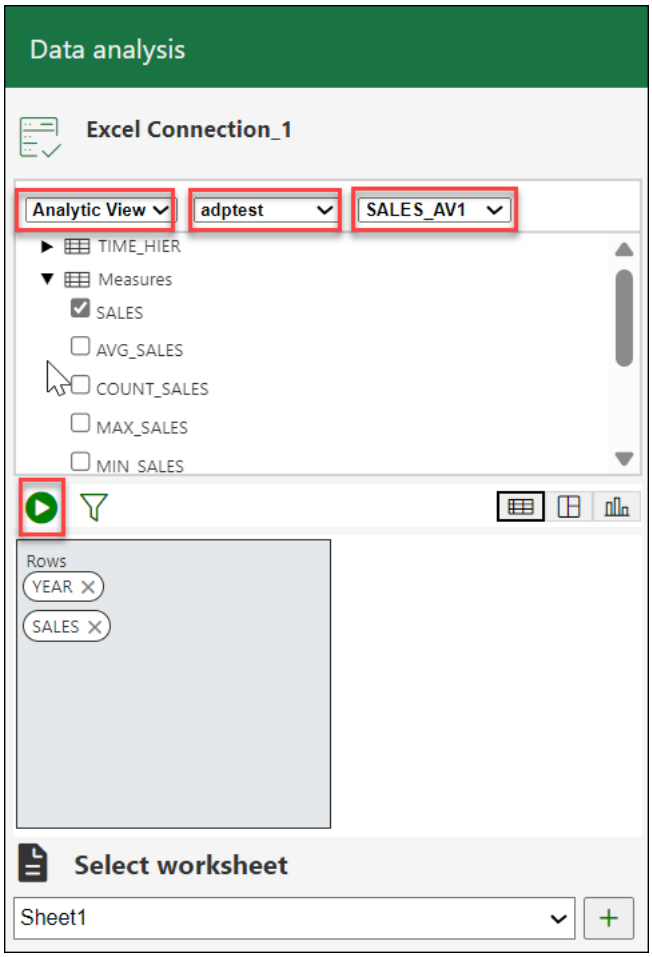
You will view the result generated in the excel worksheet.



To query an Analytic View and explore the **Data Analysis** menu in the Google Sheets:

1. Select the menu item **Autonomous Database > Data Analysis** on the Google Sheet. This opens a **Data Analysis** wizard in the Excel sheet pane.

2. Select **AV** from the **AV** or **Query** drop-down, select a schema you can access from the schema drop-down, and the AV from the available Analytic Views.

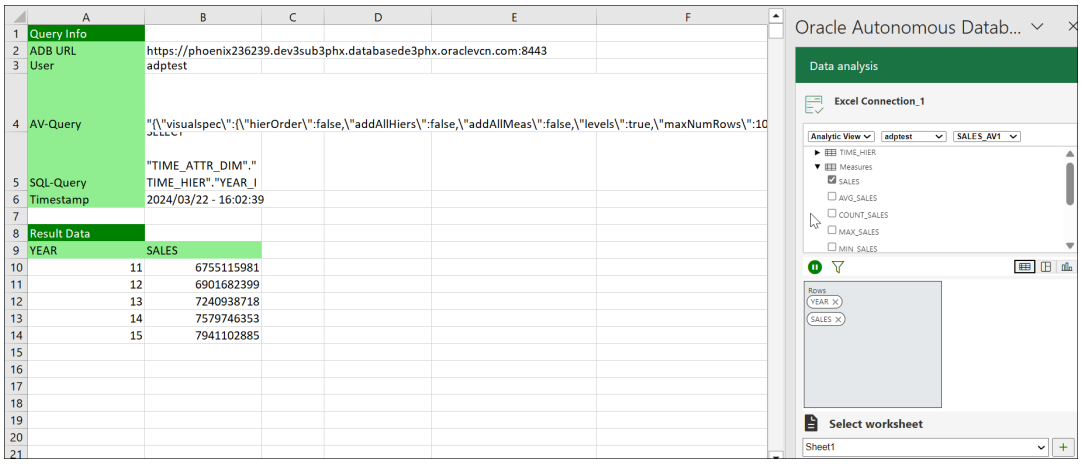


3. You can select any of the three modes to visualize the results of the AV query you generate:

   - **Table**: You can view the AV query results in tabular form. By selecting this view, a column drop zone appears, enabling you to drag and drop selected columns from the Table browser. Moving the selected columns in the drop zone allows you to view only those columns in the Result data generated in the worksheet. Select the cross mark beside the Column name to remove it from the drop zone.
   - **Pivot**: You can view the SQL query results in pivot format. By selecting this format, an X and Y drop zone appears where you can drag and drop the selected columns from the Tables browser to the drop area.
   - **Chart**: You can view the results of the AV query in chart format. By selecting this format, an X and Y drop zone appears where you can drag and drop the chosen hierarchies and measures from the AV browser to the drop area.

   > **Note:**
   >
   > You are allowed to drop measures in the Y-axis.

4. Click **Run** to view the results in the worksheet . You can view the total Sales generated along with it's year of generation.



	A	B
1	Query Info	
2	ADB URL	https://phoenix236239.dev3sub3phx.databasede3phx.oraclevcn.com:8443
3	User	adptest
4	AV-Query	"{\"visualspec\":{\"hierOrder\":false,\"addAllHiers\":false,\"addAllMeas\":false,\"levels\":true,\"maxNumRows\":10
5	SQL-Query	"TIME_ATTR_DIM"."TIME_HIER"."YEAR_I
6	Timestamp	2024/03/22 - 16:02:39
7		
8	Result Data	
9	YEAR	SALES
10	11	6755115981
11	12	6901682399
12	13	7240938718
13	14	7579746353
14	15	7941102885

FAQs for Troubleshooting errors with Excel Add-in

If you experience issues with Oracle Autonomous Database Add-in for Excel, refer to frequently asked questions in this section to identify and resolve issues.

1. Why is the **My Add-ins** icon from the **Insert** ribbon in the MS Excel workbook greyed out?

   Even before installing the Excel add-in, sometimes the **My Add-ins** icon from the **Insert** ribbon in the MS Excel workbook appears greyed out.

   a. From the **File** menu in the Excel ribbon, go to **Account** and select **Manage Settings** from the Account page.

   b. Ensure that you select the **Turn on optional connected experiences**.

   c. From the **File** menu in the Excel ribbon, go to **Options** and select the **Trust Center** option from Excel Options.

   d. Click **Trust Center Settings** and ensure that deselect **Disable all Application Add-ins** (if selected) from the Add-ins tab in the Trust Center dialog box.

   e. Select the **Trusted Add-in Catalogs** menu from the Trust Center dialog box and ensure that you deselect the **Don't Allow any web add-ins to start** checkbox.

2. Why doesn't the sign-in page of the Excel Add-in load or appear?

   Sometimes you might encounter issues with the Excel Add-in even after loading it correctly. For example, an add-in fails to load or is inaccessible. Check the compatibility version of Excel and the operating system you use.

   If the compatibility is correct and the sign-in page to the Excel Add-in still does not show up, or does not load properly, we recommend applying all pending Windows, Office, and browser updates.

   a. Select **Settings**, **Update** & **Security**, and then **Windows Update** from the Windows Start menu.

   b. If updates are available on the Windows Update page, review the updates and click **Install Now**.

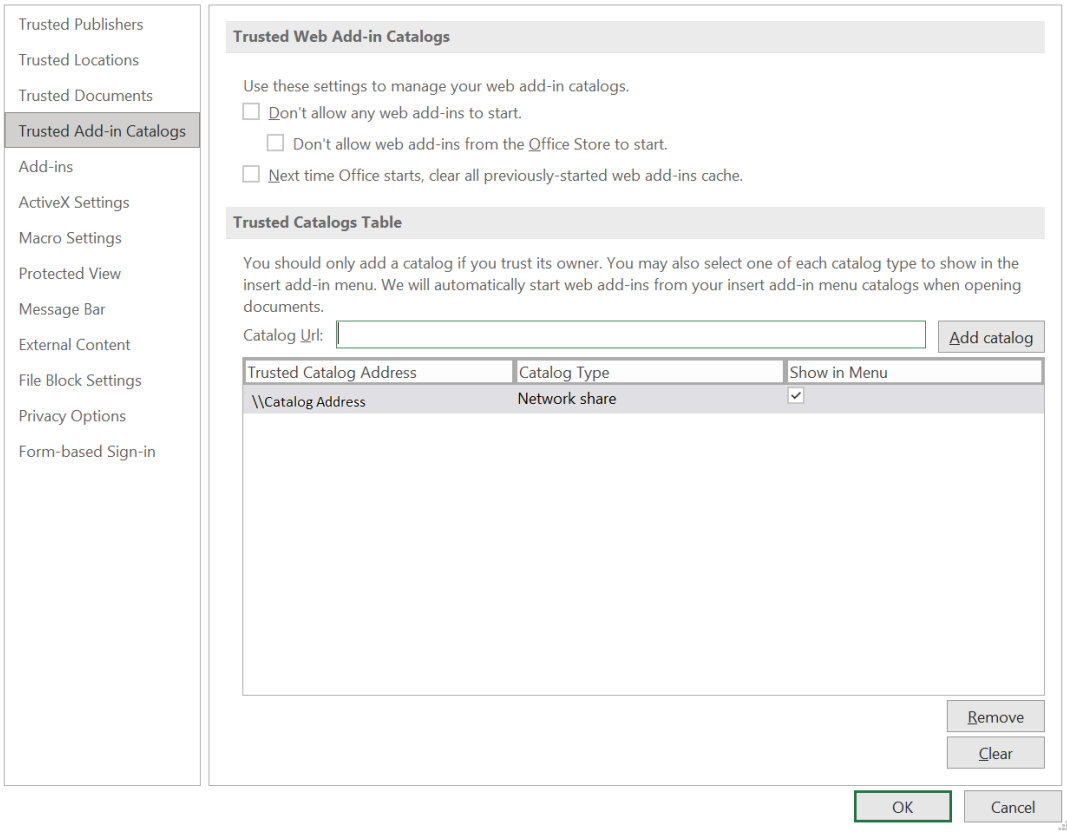
> **Note:**
>
> The details of applying Windows updates can vary from version to version and if required, check with your system administrator for assistance.

3. Why doesn't the add-in work correctly after re-installing?

   Configure the Excel trusted Add-in catalog to set the add-in correctly after re-installation.

   To configure the Excel add-in, check or remove the add-in if it is pointing at the wrong location in the Trusted catalog address. This address should be the same as the shared manifest folder's location (share path).

   Click Excel's **File> More> Options>Trust Center >Trust Center Settings> Trusted Add-in Catalogs**



   Checking is only required the first time you use the installer, or if the shared manifest folder is changed. The change occurs during uninstalling and re-pointing to a new ADB.

   To remove the catalog from the trusted table and add a new catalog pointing to a different address:

   - Select the Catalog you want to remove from the trusted catalog table and click **Remove**.
   - Enter the correct share path of the shared manifest folder in the **Catalog URL field** and click **Add Catalog** to add the shared folder to the trusted catalog.

   Restart Excel to make the new shared folder active to access the add-in.

4. Why doesn't the add-in work correctly even after configuring the Excel trusted Add-in catalog?

   Let's say you configure the Excel trusted add-in catalog after re-installing the add-in but, it does not load correctly. Sometimes the database server changes are not reflected in Excel even after you set the share path of the shared manifest folder as a trusted add-in catalog. Clear the Office cache to resolve this issue.

   Refer to this page: https://docs.microsoft.com/en-us/office/dev/add-ins/testing/clear-cache#clear-the-office-cache-on-windows to clear the Office cache on Windows and Mac.

   Clearing the Office cache unloads the Excel add-in. Install the add-in and check the configuration of the Excel trusted add-in catalog. This should solve the issue of the Excel add-in needing to be correctly loaded.

5. What should you do if you cannot view My Add-ins icon from the Excel ribbon?

   After installing the add-in once, if you cannot view the **My Add-ins** icon from the Insert ribbon and instead you view the Add-ins icon 

.

   Follow these instructions:

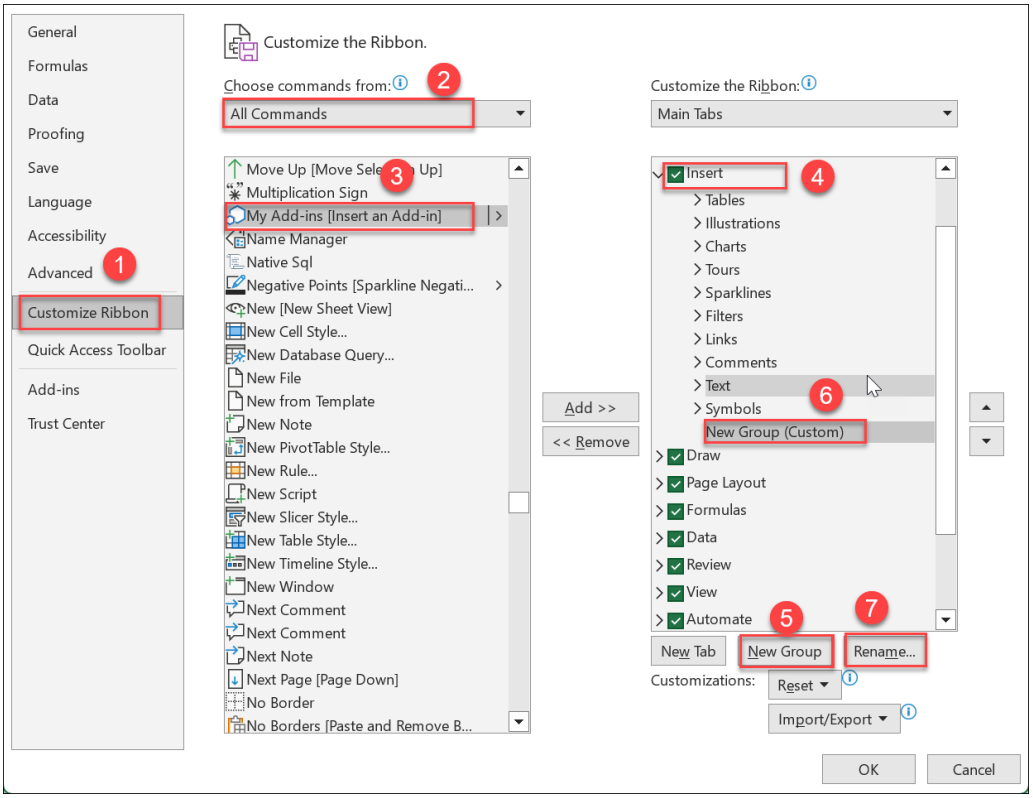
   a. From the **File** menu in the Excel ribbon, go to **Options** and select the **Customize Ribbon** option from Excel Options.

   b. Select **All Commands** drop-down from the **Choose commands from** drop-down.

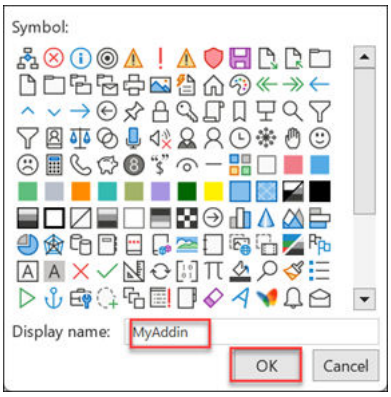
   c. Click **My Add-ins** and click **Insert** option from the Main Tabs list.

   d. Click **New Group** to add a new menu item under the **Insert** menu. You will view a **New Group (Custom)** menu option added to the Insert menu.

   e. Click **Rename** to rename the newly created menu.

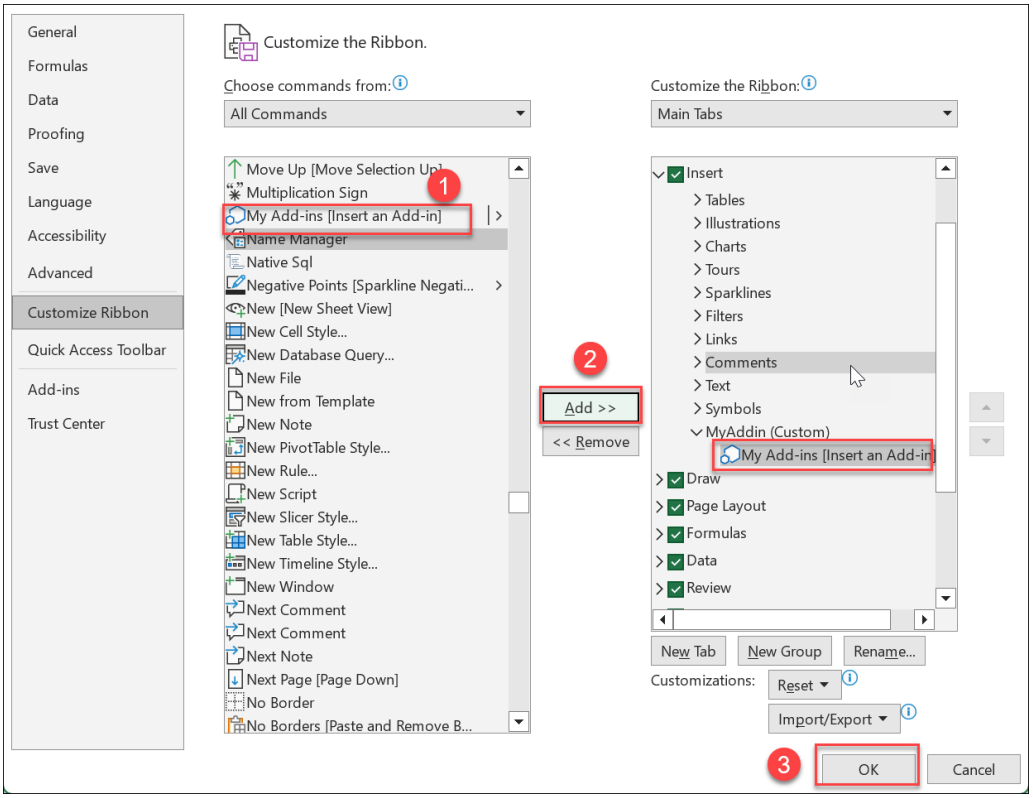


f. Specify the name of the add-in. For example, **My Addin**. Click **OK**.

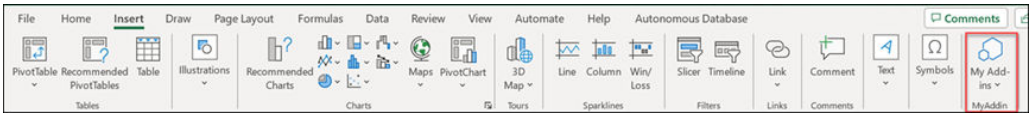


g. Click **My Add-ins[Insert and Add-in]** from All Commands list and select **Add**. My Add-ins menu is added to the newly created menu “MyAddin”.

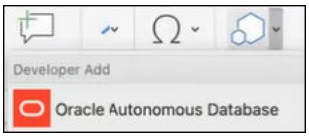
h. Click **OK** to save the changes.



i. Clicking **OK** takes you to the main Excel sheet page where you can view “**MyAdd-ins**” menu under the Insert menu.



j. Click **My Add-ins**. You can now view the Oracle Autonomous Database add-in loaded.



6. What happens when you cannot view the latest added menu items in the **Oracle Autonomous Database for Excel** add-in?

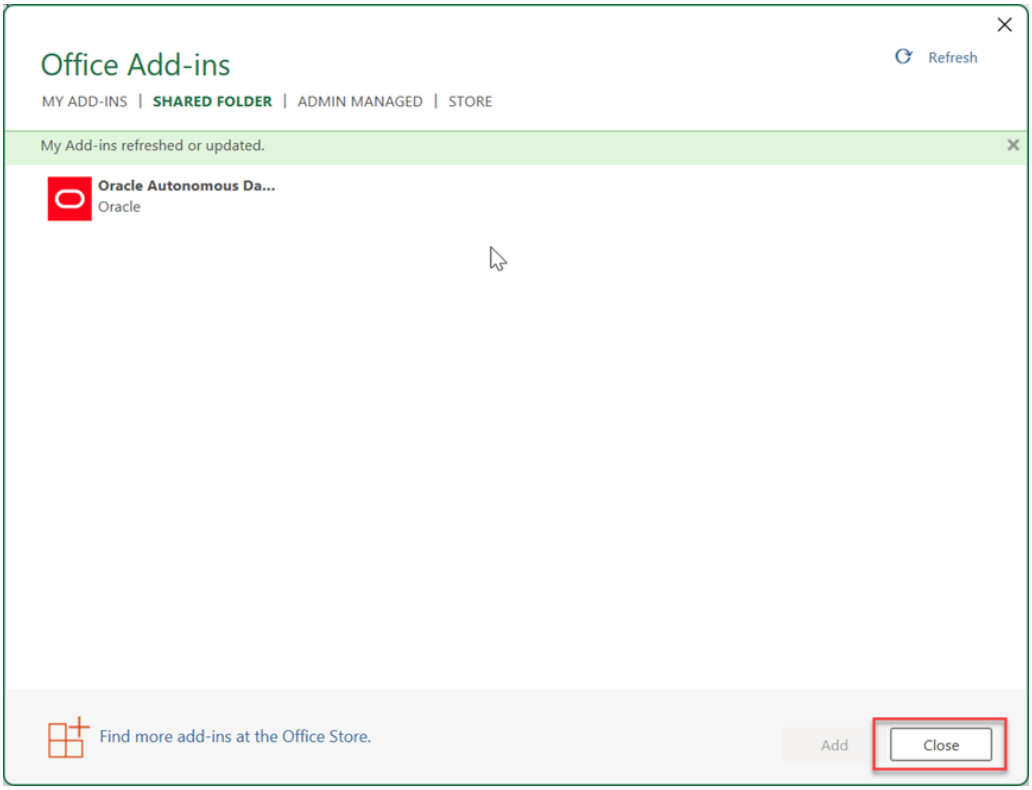
   Sometimes when you cannot view the changes updated in the latest version of the plug-in, you must:

   a. Select **My Add-ins** from the Insert menu on the Excel ribbon.

   b. Click the **Shared Folder** tab of the Office Add-ins dialog box. You will see the add-in under the Shared Folder list.

   c. Click **Refresh**.
   After the add-ins are refreshed, you will receive a notification on the dialog that says "*My Add-ins refreshed or updated*". The refresh button loads the manifest file again for the latest changes to appear.

   d. Click **Close** to close the dialog box.



Oracle Autonomous Database add-on for Google Sheets

The Oracle Autonomous Database add-on for Google Sheets enables you to query tables using SQL or Analytic Views using a wizard directly from Google Sheets for analysis.

The data retrieved from the Autonomous Database is available locally in Google Sheets for further analysis. The results are stored in the local copy and cannot be written back to the Autonomous Database. You can run direct SQL queries or query Analytic Views and view their results in the worksheet. The add-on allows you to filter the query results, and perform table joins and calculations.

> **Note:**
>
> The Oracle Autonomous Database add-on for Google Sheets must comply with Privacy Policy. For information on details of privacy policy, see Oracle Autonomous Database Privacy Policy Details.

How does the add-on for Google Sheets work?

To query an Analytic View or Tables from the Autonomous Database, you must select an Analytic View or Table to work with. While retrieving data from the Analytic View, you can configure the query according to your requirements. You can select specific hierarchies and create custom calculations on the wizard. The add-on configures your query and returns the result to the Google Sheets. You can save the results of your queries locally in the Google Sheet. The add-on can also query the schema directly to which you have access. Using the Web UI, you can also view reports and analyses you create in the Data Analysis menu in the Data Studio tool.

To use the add-on, you must enable Web Access on the Autonomous Database account. You must have the CONNECT, DWROLE, and RESOURCE roles grant in the SQL worksheet to access the Google Sheets add-on.

- Install the add-on for Google Sheets
  Before you install the Oracle Autonomous Database add-on for Google Sheets, download the *oracleGoogleAddin* zip file from your Database Actions instance.
- Upload oracleGoogleAddin files to Google Apps Script using Clasp
  To upload all the files present in the oracleGoogleAddin folder, you must use the Command Line Apps Script Project (clasp).
- Deploy the Google script as a web app
  After all the files from the oracleGoogleAddin folder are imported or uploaded to the Google Apps Script files, you must deploy the Google script as a web app.
- Download Connection File
  To connect to the Autonomous Database, you can download a connection file from the Database Actions instance and import it to the Google Sheet add-on you have setup.
- Connection Management in Google Sheets
  The Oracle Autonomous Database add-on for Google Sheets enables you to connect to multiple Autonomous Databases with a single add-on using the Connections feature. The add-on connects to Google Sheets by providing authentication to Google. Multiple users or databases can connect simultaneously to the add-on. However, only one connection can remain active.
- Generate Client ID and Client Secret using UI
  In this section you use the Web UI to obtain the `client_id` and `client_secret` .
- Authorize Google Sheets to use Autonomous Database
  After your identity is determined using OAuth authentication, Google Sheets needs permission to access the Autonomous Database.

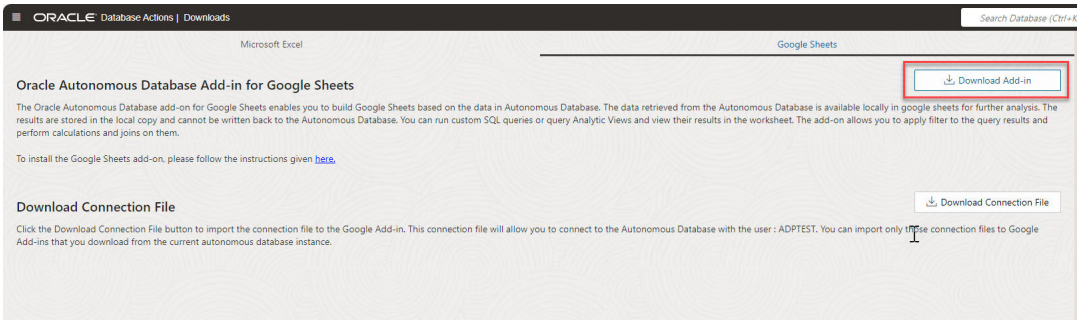
- Data Analysis in Google Sheets
  Selecting **Data Analysis** opens an Oracle Autonomous Database wizard in the Google sheet.
- Run Direct SQL Queries
  The Oracle Autonomous Database add-on for Google Sheets lets you run SQL queries to work with your data in a Google Sheet. With the add-on, you can type your SQL code in the SQL editor area and click **Run** to run the command.
- Reporting and Analysis in Google Sheets
  You can view Reports and Analytic Views or visualize data for analysis purposes.
- Clear Sheet
  Once the add-on runs the query and retrieves the data into the worksheet, you can view the Timestamp, User, AV-query and SQL-query of the Analytic View in the automatically generated query results.
- Delete all sheets
  Use this option to delete all the sheets existing in the spreadsheet.
- About menu
  Use this option to view details about the add-in
- Sign Out
  Use this option to sign out.
- Share or Publish
  Once you generate the query results in the Google Sheet, you can share it with other users. With sharing, creates a copy of the worksheet and sends it with the design tools hidden and worksheet protection turned on.
- Oracle Autonomous Database Privacy Policy Details
  This topic covers details for writing policies to control access to Autonomous Database resources.

Install the add-on for Google Sheets

Before you install the Oracle Autonomous Database add-on for Google Sheets, download the *oracleGoogleAddin* zip file from your Database Actions instance.

- Open the Database Actions Launchpad.
- Under the Downloads tab, select the **DOWNLOAD MICROSOFT EXCEL/GOOGLE SHEETS ADD-IN** pane.

- This opens a Download screen with Microsoft Excel and Google Sheets tabs. Click the Google Sheets tab and select **Download Add-in**.



You can now view a zip file in the your system's Downloads folder. Extract the contents of the zip file onto your system.

To set up the Oracle Autonomous Database add-on for Google Sheets, import the files in the *oracleGoogleAddin* folder to Google Apps Script.

> **Note:**
>
> Importing the files is a one-time activity, and typically, this is done by an administrator.

After you import or upload the files to Google Apps Script follow these steps to complete the setup of the Oracle Autonomous Database add-on for Google Sheets:

- Deploy the Google Script as a web app.
- Generate Client ID and Client Secret keys using UI.

Upload oracleGoogleAddin files to Google Apps Script using Clasp

To upload all the files present in the oracleGoogleAddin folder, you must use the Command Line Apps Script Project (clasp).

Clasp is an open-source tool to develop and manage the Google Apps Script projects from your terminal.

> **Note:**
>
> Clasp is written in Node.js. and distributed via the Node Package Manager (NPM) tool. It is required to install Node.js version 4.7.4 or later to use clasp.

1. Enter *sheet.new* in the web browser's address bar to open Google Sheets. Make sure you are logged in with your Google account.
2. Select **Apps Script** from the **Extensions** menu. You can view the Apps Script editor window.

3. Select the Code.gs file in the Apps Script editor window, which already exists by default. Click on the vertical dots beside the Code.gs file. Select **Delete** to delete the existing Code.gs file.
4. After you install Node.js, enter the following npm command in the command prompt to install clasp. You must enter this command in the location where you have downloaded and extracted the oracleGoogleAddin folder.

   ```
   C:\Users\username\Desktop\oracleGoogleAddin>npm install @google/clasp -g
   ```

   To run the command as an administrator for UNIX- and Linux-based systems, enter the following command:

   ```
   sudo npm install @google/clasp -g
   ```

   After you install Clasp, the command is available from any directory on your computer.
5. Enter the following command to log in and authorize managing your Google account's Apps Script projects.

   ```
   clasp login
   ```

   Once this command is run, the system launches the default browser and asks you to sign into your Google account where your Google Apps Script project will be stored. Select **Allow** for clasp to access your Google Account.

   > **Note:**
   >
   > If you have not enabled the Apps Script API in Google Apps Script, the above command will not be successful. Enable the API by visiting the https://script.google.com/home/usersettings site and allow site and enabling the Google Apps Script API by selecting the **On** button.

6. In your existing Google Apps Script project, click the Project Settings in your left pane. Click **Copy to Clipboard** to copy the Script ID under IDs.
7. Go back to the command prompt and enter the following command with the Script ID you copied in the previous step as displayed in the image below:

   ```
   clasp clone <Script ID>
   ```

8. Push all the files from your folder to the Google Apps Script files by specifying the following command:

   ```
   clasp push
   ```

   This command uploads all of the script project's files from your computer to Google Apps Script files.

9. Go to the newly created Google Sheet, click the Extensions menu, and select **Apps Script**. Under Files, you can view all the files in the *oracleGoogleAddin* folder.

10. After you import or upload the files to Google Apps Script, follow these steps to complete the set up of the Oracle Autonomous Database add-on for Google Sheets:

    - Deploy the Google Script as a web app
    - Generate Client ID and Client Secret keys using UI

Deploy the Google script as a web app

After all the files from the oracleGoogleAddin folder are imported or uploaded to the Google Apps Script files, you must deploy the Google script as a web app.

To deploy the Google script as a web app:

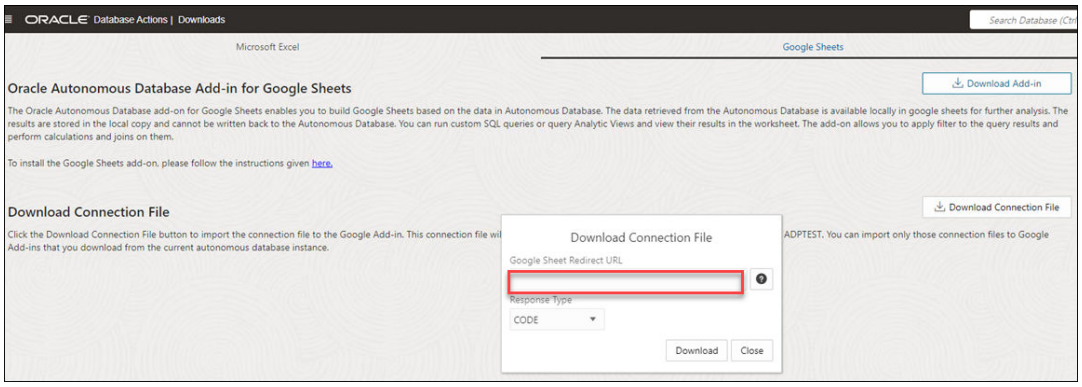
1. Click on the **Extensions** menu in the Google Sheet you are working on and select **Apps Script**. This opens the window.
2. Click **Deploy** button on the top right and select New deployment. A New deployment window opens.
3. Next to **Select type**, click the **settings** icon and select **Web app**.
4. Under Configuration, specify a Description of the deployment in the **Description** field. For example, *Web app deployment*.
5. Under **Web app** , select the Google account you used to log in from the **Execute as** drop-down. Optionally, you can choose anyone who has access to this deployment.
6. Select **Deploy**.

> **Note:**
>
> - If you receive a window that asks you to Authorize access, select it. This will redirect you to the Google Accounts page where you must to select your Gmail account.
> - Click **Advanced** and select the **Go to Untitled project (unsafe)** link.
> - Selecting the link opens a new window, ensuring that you trust the application. Click **Allow** to continue.
>
> If you Authorize access at this stage, you need not follow steps 2-4 in the *Authorize Google Sheets to use Autonomous Database*.

7. Click **Done** to close the New Deployment window.
8. Click the **Deploy** button on the top right and select **Manage Deployments**.
9. On the Manage Deployments page, you can view a Web app URL. Use the **Copy to Clipboard** to copy the Web app URL. For example, here is a sample of the web app URL

"*https://script.google.com/macros/s/AKfycbwFITvtYvGDSsrun22g7TrbrfV-bUVoWKs7OrA_3rtRAlmcGFe8bejNprZML7gFPzQ/exec*". This is the **Web application deployment URL**.

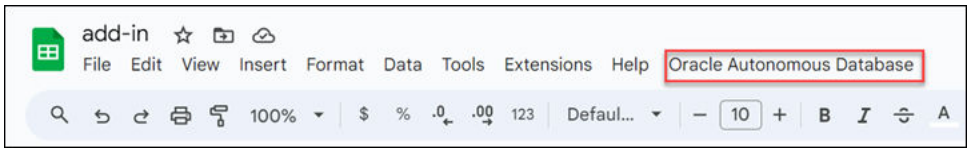
10. Save this URL, which you will use later in the **Google Sheet Redirect URL** field when downloading a connection file from Database Actions or manually creating a connection from the Google Sheet to the Autonomous Database.



For details on selecting **Response Type**, see Download Connection File.

11. You can close the Apps Script browser tab and navigate to the Google Sheets browser tab. You are now ready to create a connection to the Autonomous Database.

12. Ensure you save the worksheet after uploading all the files to Apps Script. Click the **Refresh** button once you have uploaded all the files. You can now view a new **Oracle Autonomous Database** menu in the Google Sheets.



> **Note:**
>
> Generate **OAuth Client ID** and **OAuth Client Secret** fields by using the UI.

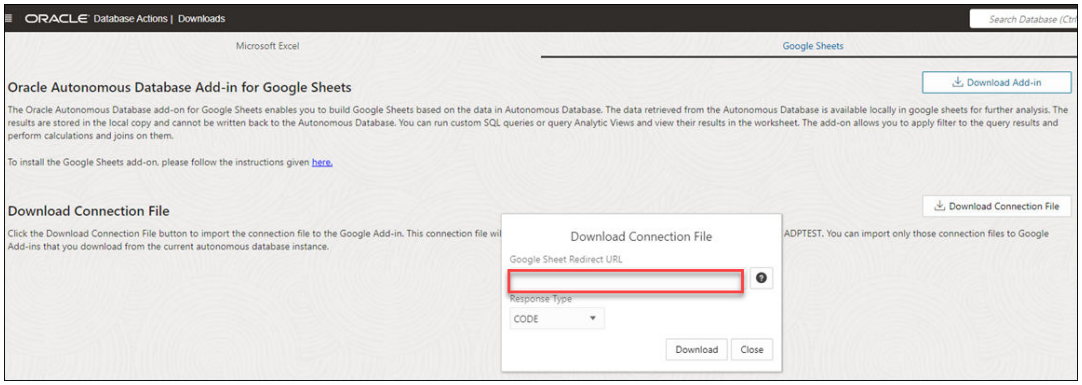
Download Connection File

To connect to the Autonomous Database, you can download a connection file from the Database Actions instance and import it to the Google Sheet add-on you have setup.

Follow the steps shown below to download the connection file.

1. Navigate to the launchpad of your Database Actions instance, and select the **DOWNLOAD MICROSOFT EXCEL/ GOOGLE SHEETS ADD-IN** Card. Click the **Download Connection File** button in the Google Sheets tab of the **Downloads** page to import the connection file to the Google Add-in.

2. This connection file will allow you to connect to the Autonomous Database with the logged-in user. You can import only those connection files to Google Add-ins that you download from the current Autonomous Database instance.



3. Selecting the Download Connection File button opens a Download Connection File wizard. Specify the following field values in the wizard:
   - **Google Sheet Redirect URL**: This is the **Web application deployment URL** you copied from step number nine of Deploy the Google Script as a Web app section.
   - Choose a **Response Type**:
     - **Explicit Connection**
       You use the OAuth Client ID and OAuth Client Secret values to authenticate and authorize Google Sheets to use the Autonomous Database. Use this when you use **CODE** as the Response Type while downloading the connection file from the Database Actions page. This is the more secure method and is preferred to use if the Autonomous database has public access.
     - **Implicit Connection**
       You will need an OAuth Client ID to implicitly access the Autonomous Database. Use this when you use Token as the **Response Type** while downloading the connection file from the Database Actions page. Use this when the autonomous database is in a private subnet or within a customer firewall.

Connection Management in Google Sheets

The Oracle Autonomous Database add-on for Google Sheets enables you to connect to multiple Autonomous Databases with a single add-on using the Connections feature. The add-on connects to Google Sheets by providing authentication to Google. Multiple users or databases can connect simultaneously to the add-on. However, only one connection can remain active.

The connection icon enables you to manage one or more database connections to databases in your Google Sheets. You can connect, edit, duplicate, and remove a connection.

You use the OAuth authentication and credentials to access the Oracle Autonomous Database for Google Sheets. The Add-on connects with the database using implicit and explicit types of connections.

Explicit Connection

You use the OAuth Client ID and OAuth Client Secret values to authenticate and authorize Google Sheets to use the Autonomous Database. Use this when you use **CODE** as the **Response Type** while downloading the connection file from Database Actions page.

Implicit Connection

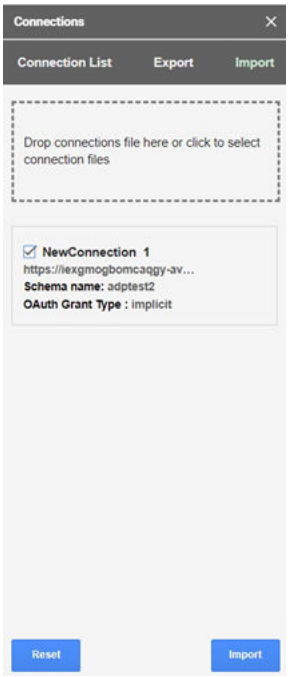
You will need an OAuth Client ID to implicitly access the Autonomous Database. Use this when you use **Token** as the **Response Type** while downloading the connection file from the Database Actions page.

Connecting to Autonomous Database

Import the connections file

Import an existing connection by clicking the **Import** tab in the Connections wizard.

1. Click the drop connection area and drag and drop the connection file saved in your local system to import the connection. You can import the **Connection File** you downloaded from the Download Connection File section.
2. Click **Import**. After you import the connection, you can view the connection in the list of connections.



3. Select the **three vertical dots** beside the connection name and click **Connect** to connect to the database.

Manual connection to the database

If you do not have a connections file to connect to the autonomous database and have access to the Schema via SQL Developer web, follow the steps outlined here to connect to the database.

Generate Client ID and Client Secret using UI

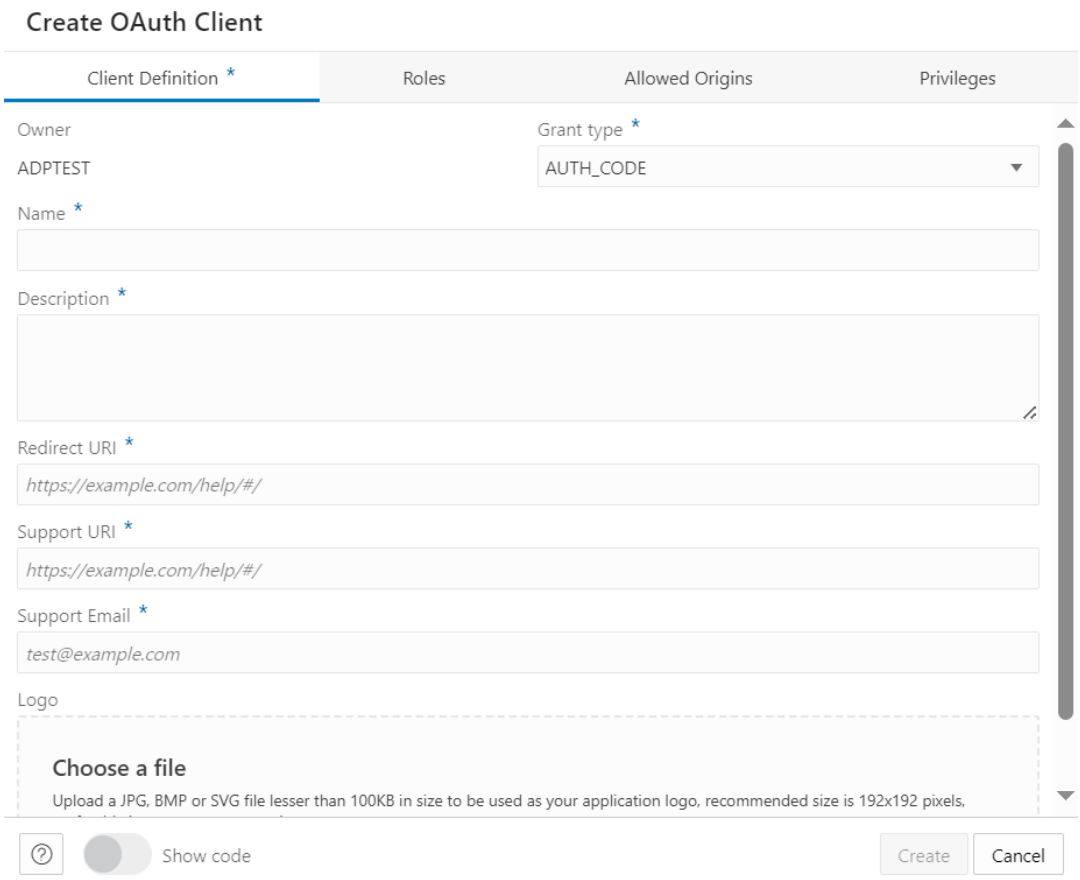
In this section you use the Web UI to obtain the `client_id` and `client_secret` .

You generate the client keys by accessing the Autonomous Database instance URL appending with oauth/clients.

For example, if your instance is “ https://<hostname>-<databasename>.adb.<region>.oraclecloudapps.com/ords/<schema Name>/_sdw/", you need to sign in to the link " https://<hostname>-

<databasename>.adb.<region>.oraclecloudapps.com/ords/<schema Name>/oauth/clients/". Be sure to include the trailing slash.

1. Sign in to Database Actions with the "https://machinename.oraclecloudapps.com/ords/SchemaName/oauth/clients/" link. You can view an OAuth Clients page in the link "https://localhost:port/ords/schemaName/_sdw/?nav=rest-workshop&rest-workshop=oauth-clients".
2. Click the **+Create OAuth Client** button to create a new client.

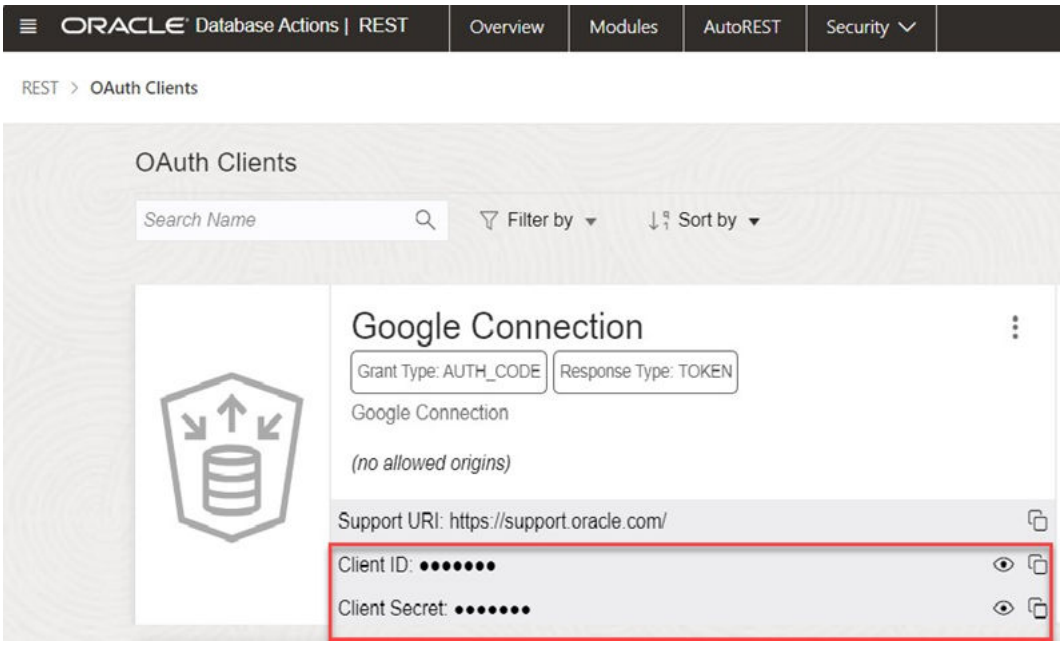


3. From the Grant type drop-down, select the type of client connection you want. You can select the following options:
   - AUTH_CODE: Select this option for implicit connection. Use this response type when the autonomous database is in a private subnet or within a customer firewall.
   - IMPLICIT: Select this option for explicit connection. This is the more secure method and is preferred to use if the Autonomous database has public access.
4. Enter the following fields. The fields with an asterisk (*) are mandatory:
   - **Name**: Name of the client.
   - **Description:** Description of the purpose of the client.
   - **Redirect URI**: web application deployment URL you copied from step 10 of Deploy the Google Script as a Web app
   - **Support URI**: Enter the URI where end users can contact the client for support. Example: *https://script.google.com/*

- **Support Email**: Enter the email where end users can contact the client for support.
- **Logo**: Optionally, select an image from your local system to insert a logo for your new client.

Navigate to the **Roles** tab to select the roles of the client. This is not a mandatory field.

5. Progress to the **Allowed Origins** tab. Specify and add the list of URL prefixes in the text field. This is not a mandatory field.
6. Progress to the **Privileges** tab to add any privilege. You are not required to have any privileges to create an OAuth Client.
7. Click **Create** to create the new OAuth Client. This registers the OAuth Client which you can view on the OAuth Clients page.

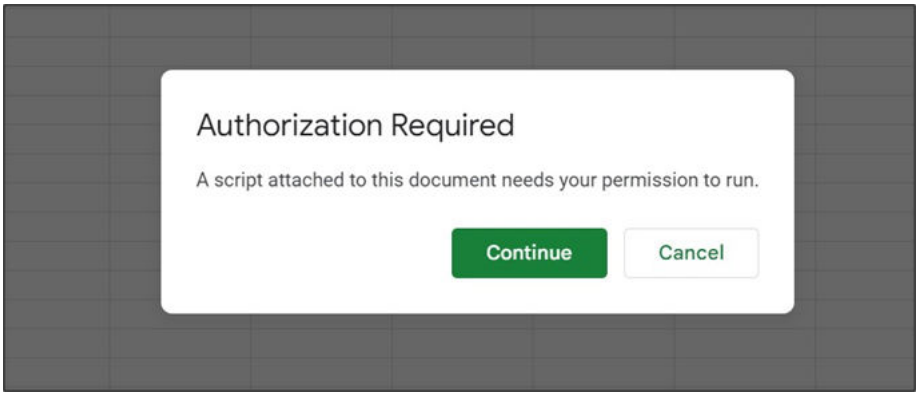


8. Click the show icon to view the `Client ID` and the `Client Secret` fields.

How do I connect manually?

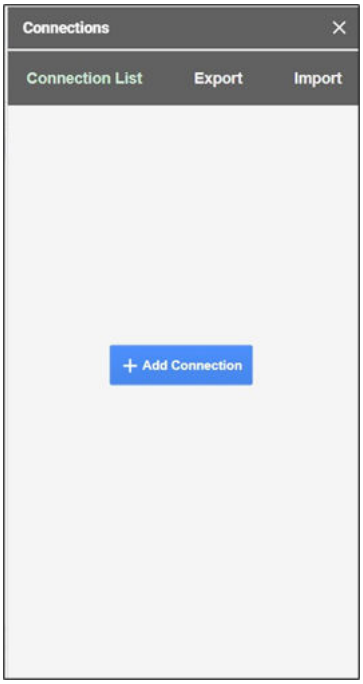
The following sections demonstrate how to connect using implicit and explicit connections. Google Sheets needs permission to access the Autonomous Database. You must first complete the authorization to connect to the autonomous database. The add-on requires one-time authentication for the setup.

1. On the Google Sheet, click **Oracle Autonomous Database** and select **Connections**. Selecting Connections requires one-time Google authentication.
   - Clicking **Connections** opens a pop-up window that asks your permission to run the authorization. Click **Continue**.

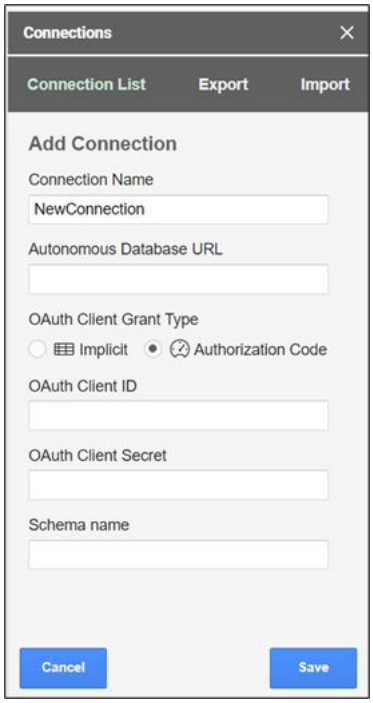


- You will now view a window that informs you that the application requests access to sensitive information in your Google account.
- Click **Advanced** and select the **Go to Untitled project (unsafe)** link. Selecting the link opens new window, ensuring you trust the application. Click **Allow** to continue. You have now completed the setup.

2. On the Connections wizard, click **Add Connection** to add a connection.



3. Selecting Add Connection opens an **Add Connection** wizard in the Connections wizard's connection list panel.



4. Specify the following field values in the wizard:

   Connection Name: Enter the connection's name—for example, *TestConnection*.

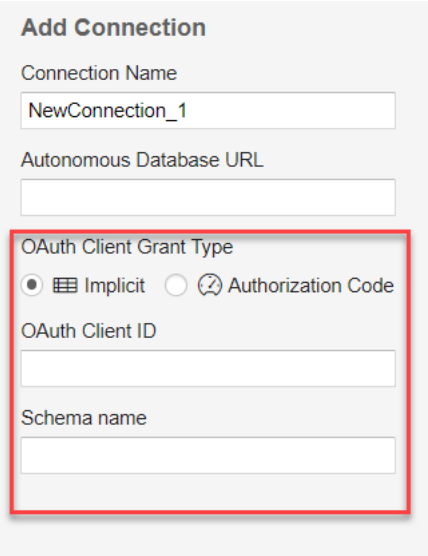
   Autonomous Database URL: Enter the URL of the Autonomous Database you wish to connect to. For example, "*https://<hostname>-<databasename>.adb.<region>.oraclecloudapps.com/*"

   In the OAuth Client Grant Type field, select one of the two options based on the type of connections you want. Refer to the Generate Client ID and Client Secret using the UI section.

   This option varies with implicit and explicit connections.

   Implicit: Select this option for implicit connection. Use this response type when the autonomous database is in a private subnet or within a customer firewall.
   Authorization Code: Select this option for explicit connection. This is the more secure method and is preferred to use.

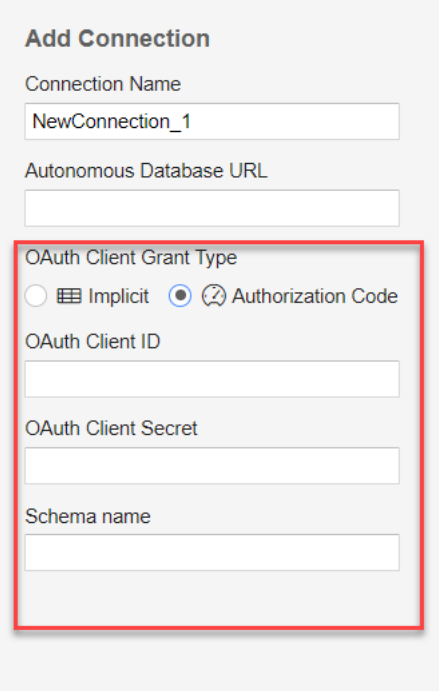
   When you select the Implicit option, you can view the following fields:



OAuth Client ID: `client_id` you generate using the *Create New Client* wizard in the UI. Refer to the Generate Client ID and Client Secret using the UI section.

Schema Name: Specify the name of the schema.

When you select Authorization Code, you can view the following fields:



OAuth Client ID: `client_id` you generate using the Create New Client wizard in the UI. Refer to the *Generate Client ID and Client Secret using the UI* section.

OAuth Client Secret: `client_secret` you generate using the Create New Client wizard in the UI. Refer to the *Generate Client ID and Client Secret using the UI* section.

Schema Name: Specify the name of the schema.

Click **Save**.

After you click **Save**, you can view the new connection in the connection list panel. The connection list displays the connection's name, the schema's name, and the OAuth type you grant. However, it is still in a disconnected state.

5. Click the three vertical dots beside the connection name and perform the following operations:

   Connect: Select **Connect** to the Autonomous Database and change the connection status to active. Selecting **Connect** opens the sign-in page of the Autonomous database. After you log in, you will view a page that shows that database access has been granted to you. Close the window and return to Google Sheets. You will now see that the connection is active.

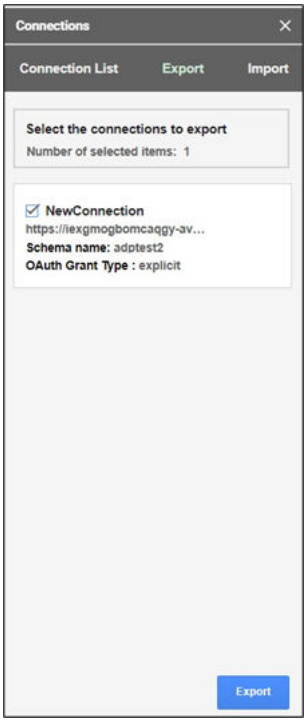
   Edit: Select **Edit** to update any value of the connection. Click **Save** to update the edited values.

   Duplicate: Select **Duplicate** to create a duplicate connection.

   Remove: Select **Remove** to remove the connection from the connection list.

Exporting Connections

1. Click the **Export** tab in the Connections wizard to export the selected connection.
2. Select the connection you want to export, and click **Export**.



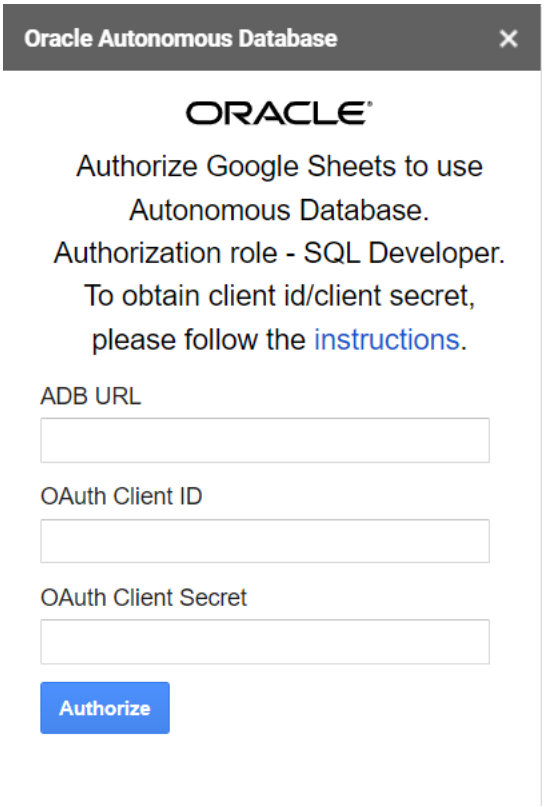
3. Click **Export**.
4. The exported connection downloads in your local system. The connection file is saved as *spreadsheet_addin_connections.json*.

Authorize Google Sheets to use Autonomous Database

After your identity is determined using OAuth authentication, Google Sheets needs permission to access the Autonomous Database.

The `client_id` and `client_secret` values you generate during OAuth authentication are used for authorization.

1. Click on the **Oracle Autonomous Database** menu in the Google Sheet you are working on and select **Register**. This requires one-time Google authentication.
2. Clicking Register opens a pop-up window that asks your permission to run the authorization. Click **Continue**. Selecting Continue will redirect you to the Google Accounts page, where you must select your Gmail account.
3. You will now view a window that informs you that the application requests access to sensitive information in your Google account. Click Advanced and select the Go to Untitled project (unsafe) link.
4. Selecting the link opens a new window, ensuring you trust the application. Click **Allow** to continue.
5. You have now completed the setup. Select **Register** from the Oracle Autonomous Database menu in the Google sheet.
   This opens an Oracle Autonomous Database wizard in the Google sheet. Specify the following fields:
   - **ADB URL**: Enter the ADB URL. For example, "*https://<hostname>-<databasename>.adb.<region>.oraclecloudapps.com/ords/<Schema Name>*".
   - **OAuth Client ID**: client_id you generate during authentication.
   - **OAuth Client Secret**: client_secret you generate during authentication. Refer to the Create Connections with the Google spreadsheet section for more details.



6. Select **Authorize**.

After successfully authorizing the credentials, you can view **Connections**, **Direct SQL**, **Data Analysis**, **Analyses and Reports Clear Sheet**, **Delete All Sheets**, **About Autonomous Database**, and **Sign Out** menu items under **Oracle Autonomous Database**.

Data Analysis in Google Sheets

Selecting **Data Analysis** opens an Oracle Autonomous Database wizard in the Google sheet.

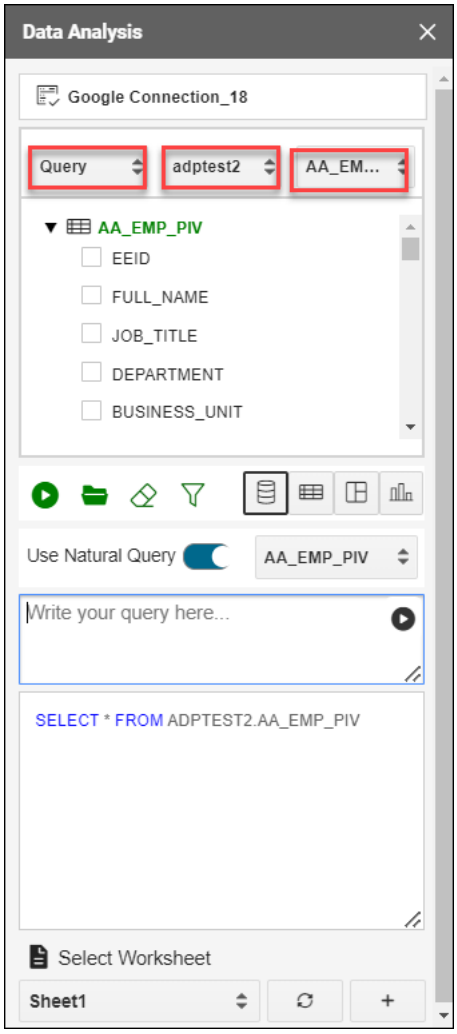
The add-on enables you to receive a copy of data from the Autonomous Database to the Google sheet. You can query an existing Analytic View and run SQL Query using the Oracle Autonomous Database Wizard.

You can retrieve the Analytic View and manipulate the query according to your requirements to visualize the result data in the worksheet. You can search for the Analytic View and select measures, hierarchies, and levels from the query. You can also add filters and calculated measures to the query and view the result in the sheet. By default, the data is retrieved in tabular format.

You can run custom queries. The add-on enables you to apply a filter to the query results. The add-on lets you to view query results that can be customized with selected columns using a faceted filter.

To run a custom query using the add-on:

1. On the Google Sheet, select the menu item **Oracle Autonomous Database**.
2. Select **Data Analysis**. Selecting Data Analysis opens a Data Analysis wizard. On the Data Analysis wizard, select **Query** from the drop-down, the **schema** you wish to use from the drop-down, and the **table** on which you perform the query.

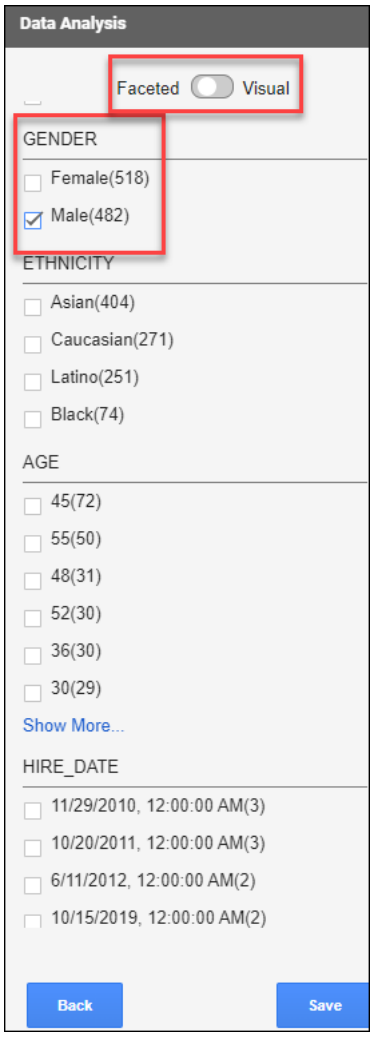


3. You will view the default query in the query editor area. You can select any of the four modes to visualize the results of the SQL query report you generate:

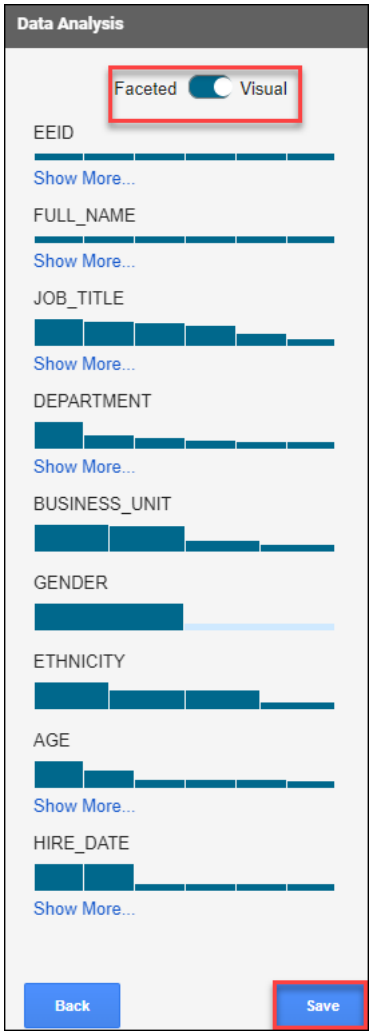
- **Base Query**: This type of view is by default. The query written in the SQL editor is the Base Query.
- **Table**: You can view the SQL results in tabular form. By selecting this view, a column drop zone appears, enabling you to drag and drop selected columns from the Table browser. Moving the selected columns in the drop zone allows you to view only those columns in the Result data generated in the worksheet. Select the cross mark beside the Column name to remove it from the drop zone.
- **Pivot**: You can view the SQL query results in pivot format. By selecting this format, an X and Y drop zone appears where you can drag and drop the selected columns from the Tables browser to the drop area.
- **Chart**: You can view *Area Chart*, *Bar Chart*, *Line Chart*, or *Pie Chart* when you select this option.. The mappings displayed when you select one of the options are as follows:

- **Orientation**: Choose between horizontal and vertical orientation types from the drop-down list.
- **X axis label and Y axis label**: Optionally enter labels for X axis and Y axis.

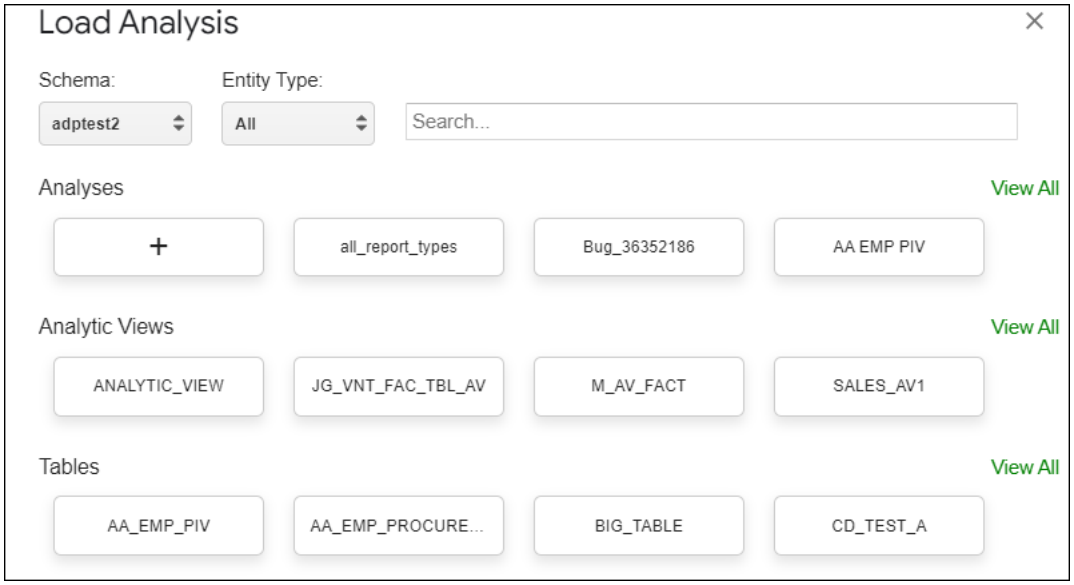
4. Click the **funnel icon** (Faceted Filter) to add filters to the result. The wizard generates a filter for each value in the column retrieved from the query result. You can filter different columns on the faceted filter panel and view the results in the worksheet to view only the data you wish to view. For example, to view the customer reports by Gender, click the **faceted filter** and select **Male** under Gender.



Use the Visual facet: Use the visual indicator to graphically represent the faceted filter.



5. Click **Load Analysis** to view the list of Analyses, Analytic Views, and Tables from the schema you select. You can select **All**, **Analysis**, **Analytic View** and **Table** from the **Entity Type** drop-down.

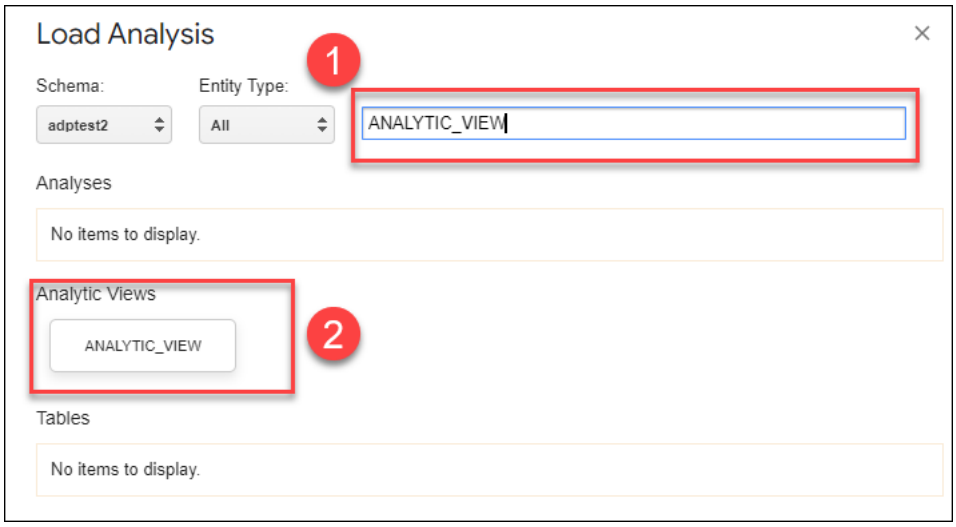


The list displayed on the dialog box varies with the choice you make with the type of entity.

Click **View All** to view all the entities present in the schema.

Selecting an Analysis will also display the Reports associated with the selected Analysis.

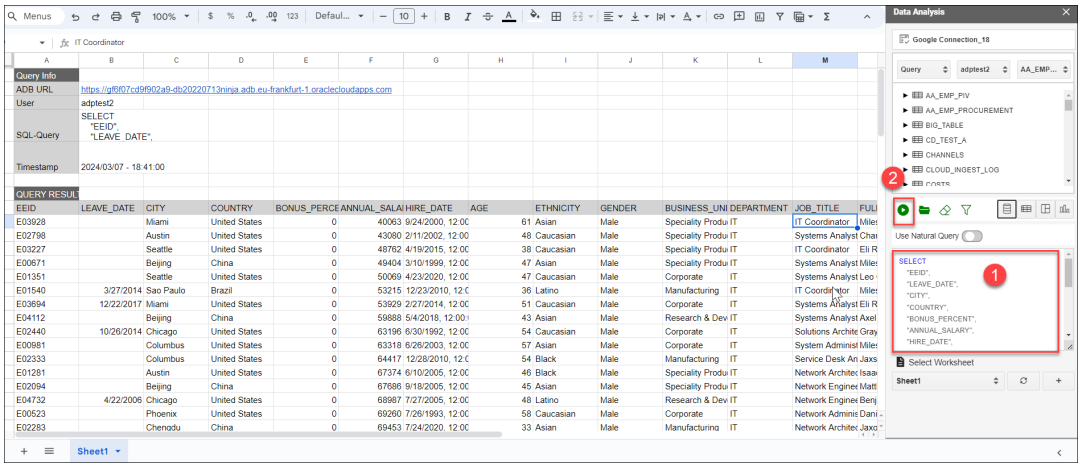
Click the search field and start typing the name of the Analysis, Analytic View, or Table you are looking for. For example, if you are looking for an Analytic View named `ANALYTIC_VIEW`.



You will view the selected search entity below the search field.

Click `X` to close the **Load Analysis** dialog.

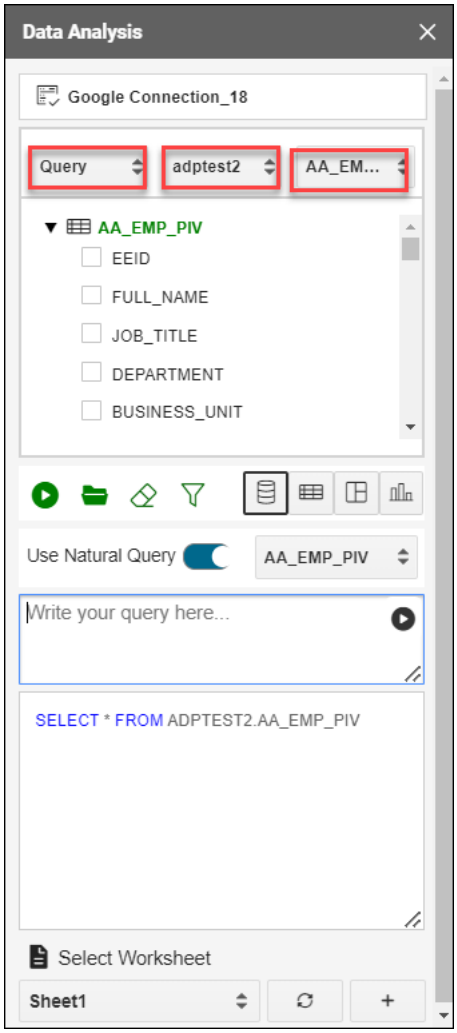
6. Select **Run** to generate the results of the custom query in the worksheet.



Using natural language to interact with your database data is now achievable with Oracle Autonomous Database add-on for Google Sheets. This means you can use natural language, for example, plain English, to query the database. This means you can provide a natural language prompt instead of SQL code to interact with your data. When you select **Use Natural Query** the add-on converts natural language to SQL.

To run a natural query using the add-on:

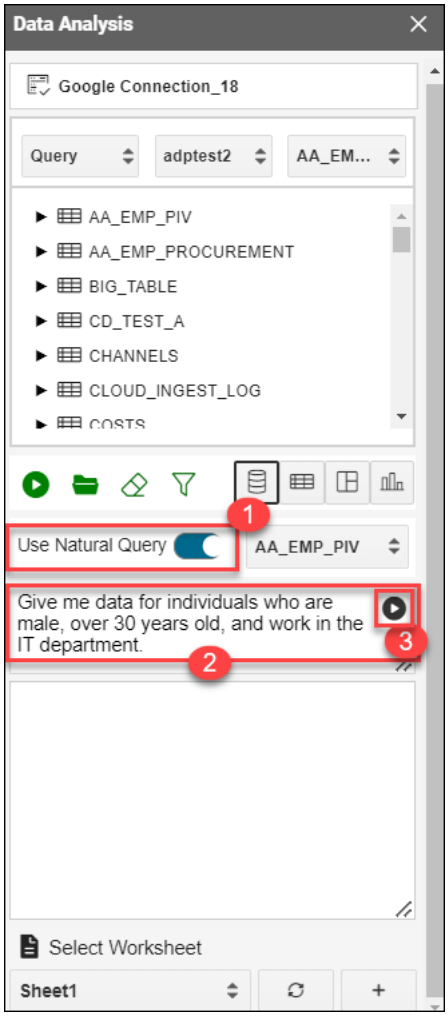
1. On the Google Sheet, select the menu item **Oracle Autonomous Database**.
2. Select **Data Analysis**. Selecting Data Analysis opens a Data Analysis wizard. On the Data Analysis wizard, select **Query** from the drop-down, the **schema** you wish to use from the drop-down, and the **table** on which you perform the query.



3. Select **Use Natural Query**. Let's say you want details of employees who are male, over 30 years old and work in the IT department. Enter the following natural language query in the query display area:

```
Give me data for individuals who are male, over 30 years old, and work in
the IT
      department.
```

4. Click **Generate Query** to produce the equivalent SQL query in the bottom query display area.

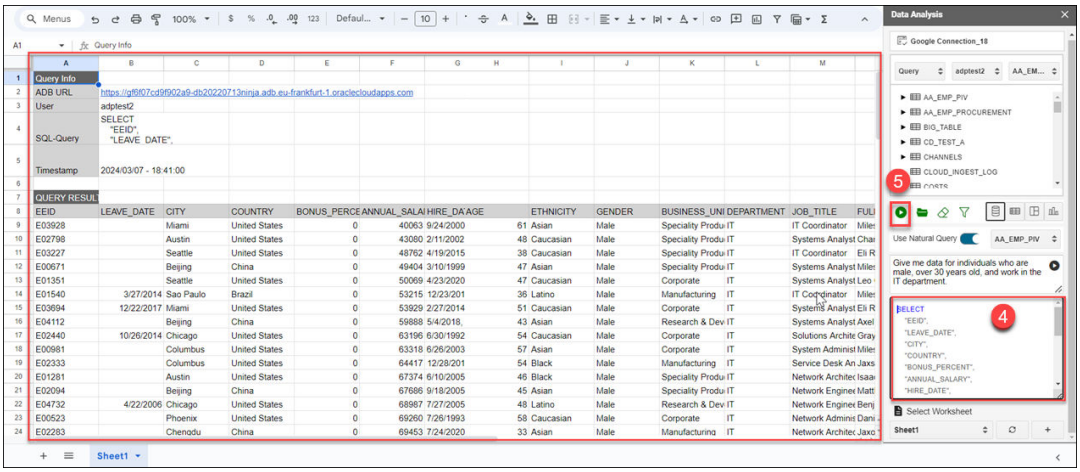


5. You will view the following code in the bottom SQL code area.

```
SELECT
    "EEID",
    "LEAVE_DATE",
    "CITY",
    "COUNTRY",
    "BONUS_PERCENT",
    "ANNUAL_SALARY",
    "HIRE_DATE",
    "AGE",
    "ETHNICITY",
    "GENDER",
    "BUSINESS_UNIT",
    "DEPARTMENT",
    "JOB_TITLE",
    "FULL_NAME"
FROM
    "ADPTEST2"."AA_EMP_PIV"
WHERE
        "GENDER" = 'Male'
```

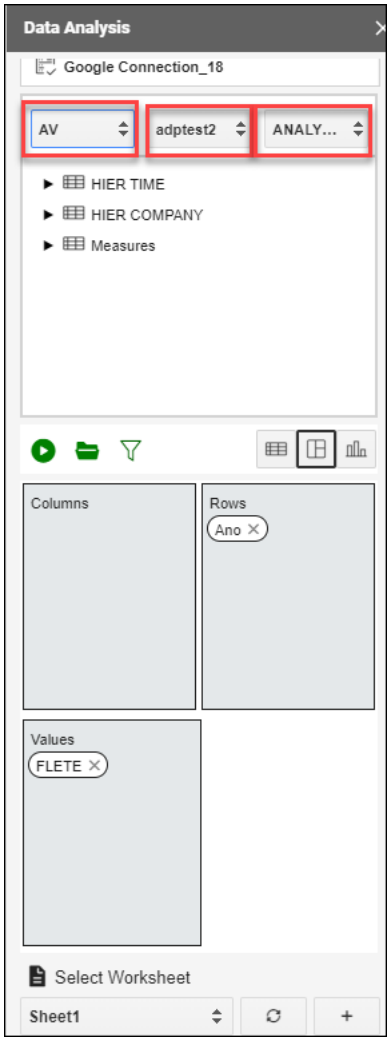
```
AND "AGE" > 30
AND "DEPARTMENT" = 'IT'
```

6. Click **Run** to run the query and display the results in the worksheet. You can click the **+** sign beside the **Select worksheet** drop-down to display the results in a new worksheet.



To query an Analytic View and explore the **Data Analysis** menu in the Google Sheets:

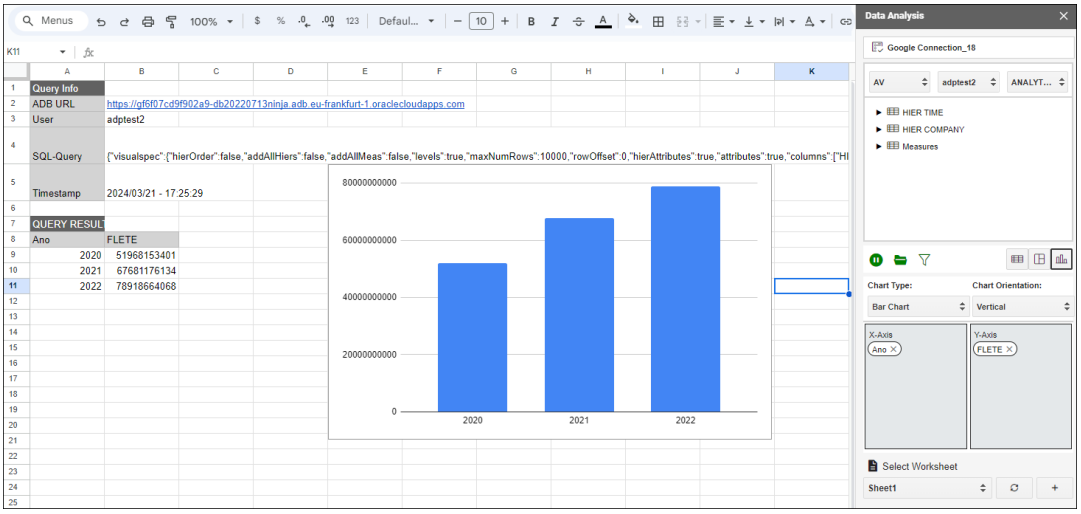
1. Select the menu item **Oracle Autonomous Database > Data Analysis** on the Google Sheet. This opens a **Data Analysis** wizard in the Google task pane.
2. Select **AV** from the **AV** or **Query** drop-down, select a schema you can access from the schema drop-down, and the AV from the available Analytic Views.



3. You can select any of the three modes to visualize the results of the AV query you generate:
   - **Table**: You can view the AV query results in tabular form. By selecting this view, a column drop zone appears, enabling you to drag and drop selected columns from the Table browser. Moving the selected columns in the drop zone allows you to view only those columns in the Result data generated in the worksheet. Select the cross mark beside the Column name to remove it from the drop zone.
   - **Pivot**: You can view the SQL query results in pivot format. By selecting this format, an X and Y drop zone appears where you can drag and drop the selected columns from the Tables browser to the drop area.
   - **Chart**: You can view the results of the AV query in chart format. By selecting this format, an X and Y drop zone appears where you can drag and drop the chosen hierarchies and measures from the AV browser to the drop area.

   > **Note:**
   >
   > You are allowed to drop measures in the Y-axis.

4. Click the **funnel** icon to view the Faceted and Visual Filter list. The wizard generates a filter for each value in the column retrieved from the query result. You can filter different columns on the faceted filter panel and view the results in the worksheet to view only the data you wish to view.
5. Click **Save**.
6. Click **Run** to view the results in the worksheet you select consisting of the Drama genre.



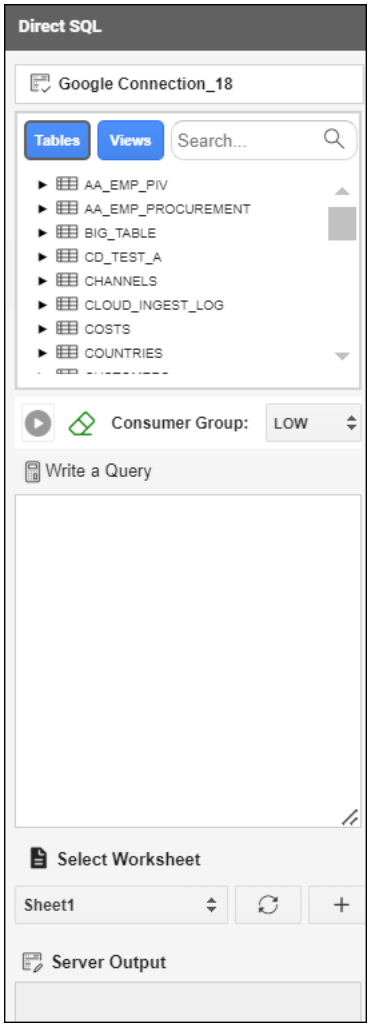
Run Direct SQL Queries

The Oracle Autonomous Database add-on for Google Sheets lets you run SQL queries to work with your data in a Google Sheet. With the add-on, you can type your SQL code in the SQL editor area and click **Run** to run the command.

The add-on loads the result in the Google Sheet. The time taken to load the results depends on the number of records and the complexity of the query.

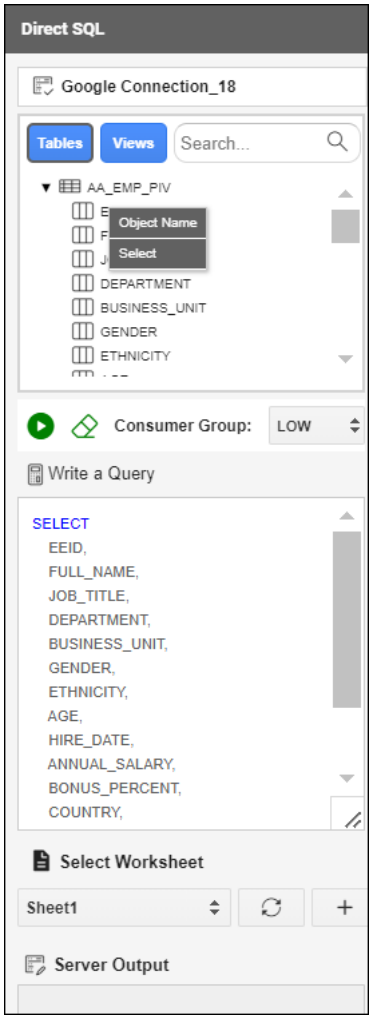
To run a query using the add-on, open Google Sheets and a blank workbook.

1. In the Google Sheet, select the menu item **Oracle Autonomous Database**.
2. Select **Direct SQL** to type and run the SQL command.
3. The Oracle Autonomous Database wizard opens Tables and Views icons and a search field beside it.

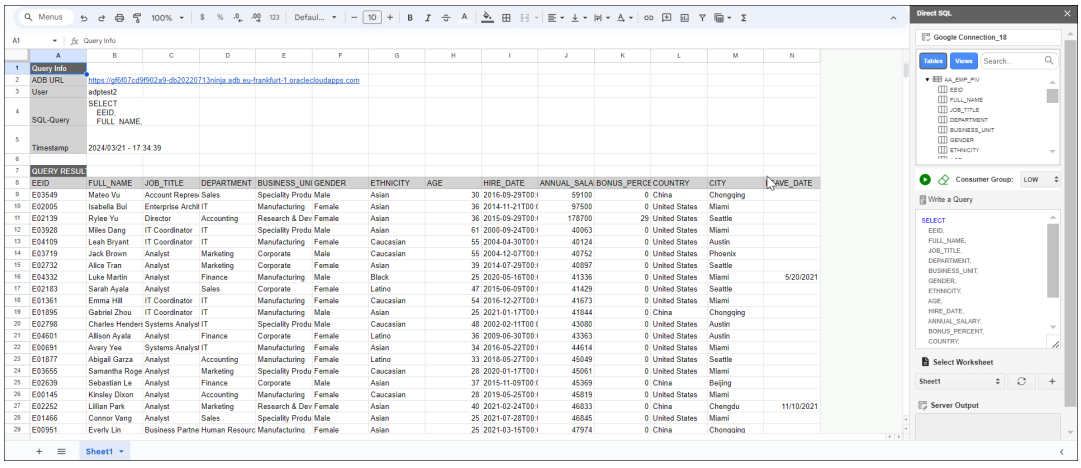


4. Select **Table** to view all the tables in the database. Perform the same operations for Views.
5. You can right-click on the table whose data you want to query and choose **Select** to view all the table's columns. The column names will be displayed in the **Write a Query** section. You can click on the table and view individual columns as well.

> **Note:**
>
> You can also select the consumer group options such as, low, medium, and high. You can edit the existing query from the query editor.



6. Click **Run** to run the query and display the results in the worksheet. You can click the **+** sign beside the **Select worksheet** drop-down to display the results in a new worksheet.
7. The worksheet also displays the timestamp, the user who creates and runs the query, the ADB URL, and the SQL Query.



Reporting and Analysis in Google Sheets

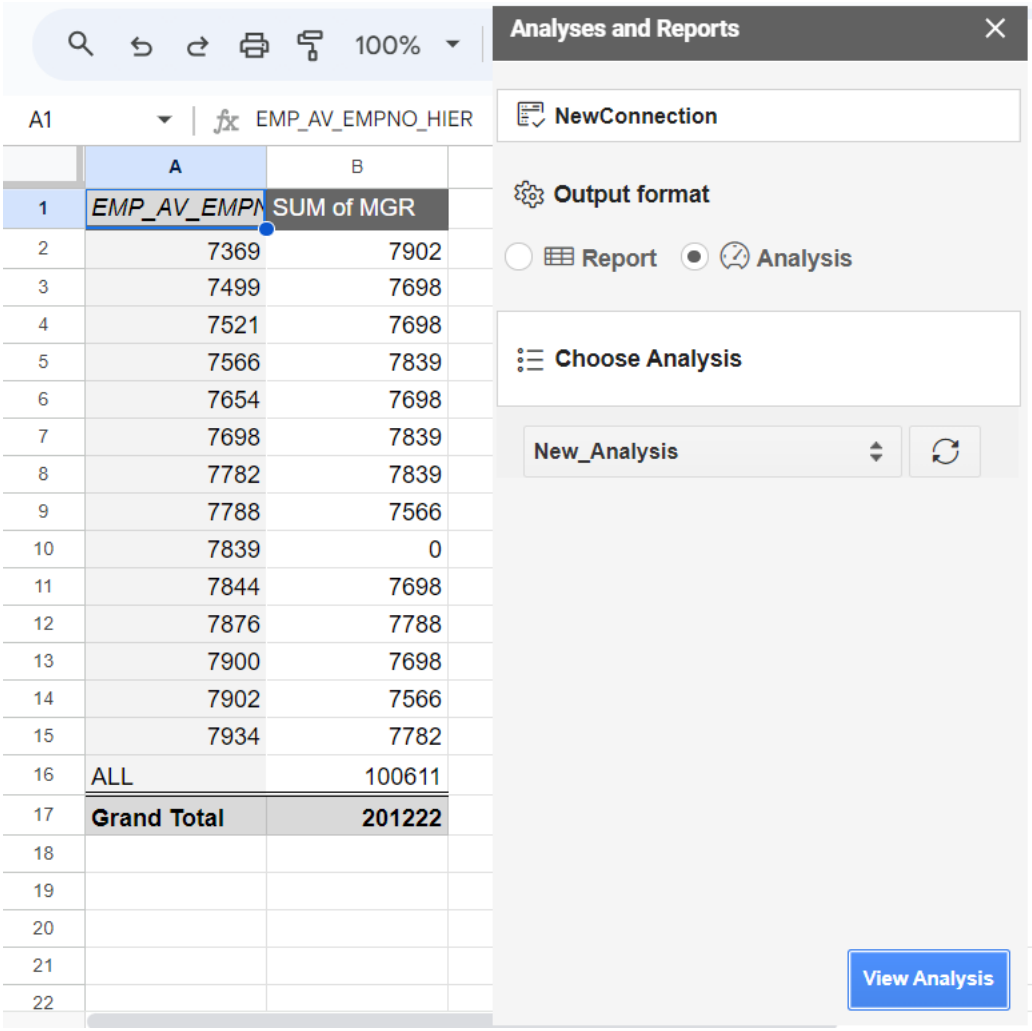
You can view Reports and Analytic Views or visualize data for analysis purposes.

The reports and charts can be viewed in various charts: Bar Charts, Area charts, Line Charts, and Pie Charts. Reports provide analytical insights that you create from the Analytic Views. An Analysis can contain multiple reports. The Analyses and Reports icon enables you to retrieve Analyses and Reports from the Autonomous Database.

View Analysis

To view Analysis and explore the Analyses and Reports menu:

1. Select **Analysis** under Output format.
2. Use the Select an Analysis drop-down to choose the Analysis you want to view.
3. Click **View Analysis** to view the analysis in the Google Sheet.

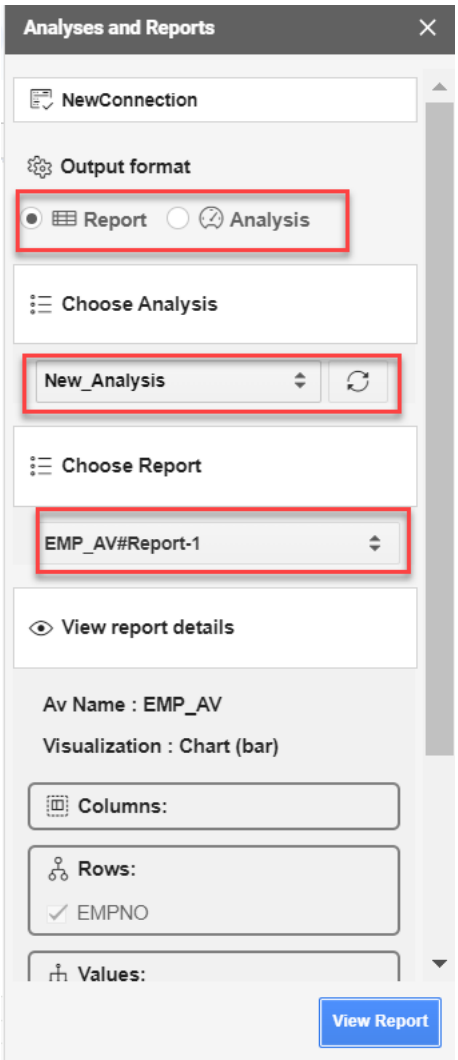


	A	B
1	EMP_AV_EMPN	SUM of MGR
2	7369	7902
3	7499	7698
4	7521	7698
5	7566	7839
6	7654	7698
7	7698	7839
8	7782	7839
9	7788	7566
10	7839	0
11	7844	7698
12	7876	7788
13	7900	7698
14	7902	7566
15	7934	7782
16	ALL	100611
17	**Grand Total**	**201222**
18		
19		
20		
21		
22		

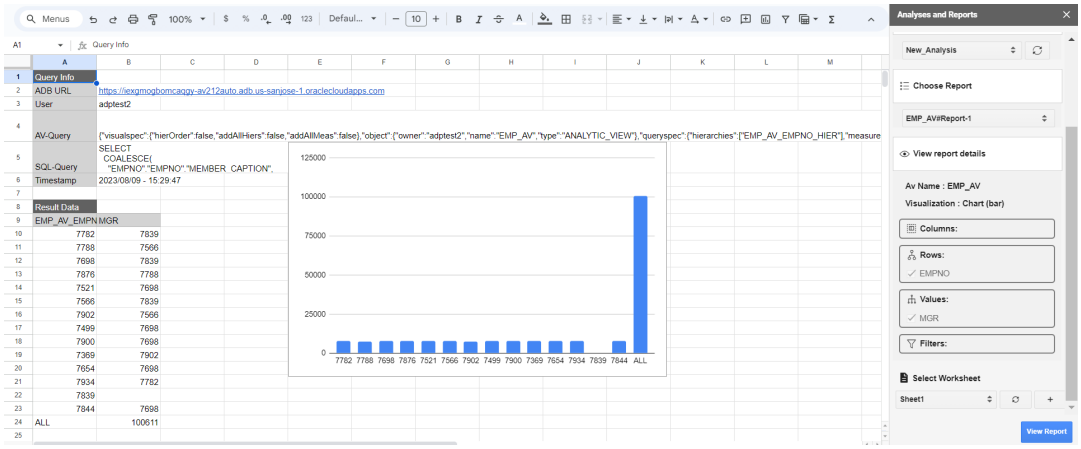
View Report

To view Reports :

1. Select the **Analyses and Reports** menu from the **Oracle Autonomous Database** menu. This opens the Analyses and Reports wizard.
2. Select **Report** under Output Format.
3. Use the **Select an Analysis** drop-down under **Choose Analysis** to choose the Analysis you want to view.
4. After you select the Analysis, to view the report present in the Analysis, click the **Select a report** drop-down and select the report you wish to view.
5. Click **View Report Detail** to view more information about the report: Analytic View Name, Type of visualization, and rows, columns, and values you select while creating the report.



6. Select the worksheet from the drop-down where you would want to view the report.
7. Click **View Report** to view the report in the selected sheet from the previous step. You can now view the report in the worksheet you select.



Clear Sheet

Once the add-on runs the query and retrieves the data into the worksheet, you can view the Timestamp, User, AV-query and SQL-query of the Analytic View in the automatically generated query results.

Once the add-on runs the query and retrieves the data into the worksheet, you can view the Timestamp, User, AV query, and SQL query of the Analytic View in the automatically generated query results.

The worksheet displays the result of the query in one go. Consider, for example, if you want to modify the query and generate the query result in the same sheet. You must clear the existing data in the sheet.

To clear query results in the Google sheet, click the menu item **Oracle Autonomous Database** and select **Clear Sheet**.

This option erases all data types in the selected sheet, including images and formatting.

Delete all sheets

Use this option to delete all the sheets existing in the spreadsheet.

Select **Delete All Sheets** from the **Oracle Autonomous Database** menu to delete all sheets from the spreadsheet.

About menu

Use this option to view details about the add-in

The **About** menu from **Ask Oracle** displays if the add-on is connected to server, ORDS version, the add-on version, ORDS Schema version and the database version.

Sign Out

Use this option to sign out.

Select **Sign Out** menu from **Oracle Autonomous Database** to sign out from the database session.

Share or Publish

Once you generate the query results in the Google Sheet, you can share it with other users. With sharing, creates a copy of the worksheet and sends it with the design tools hidden and worksheet protection turned on.

The recommended steps to take before you publish are:

1. Review and inspect to remove personal or sensitive information.
2. Save the source version of the worksheet. Consider adding a file name suffix of –src for the source worksheet. Then, remove the suffix in the distributed copy.
   Once you are ready to distribute to the users, click **Share**.
   - In the Share window that appears, add the user email IDs with whom you want to share the Sheets and to whom you want to provide permissions for accessing the Sheets.
   - You can select the permission of the users from the drop-down. Select **Editors** if you want the user to share the worksheet. **Viewers** and **commenters** can see the option to download, print and copy but not share the sheets.
   - Select **Notify people** check-box to notify the users of the share.
   - Under General access, select **Restricted** from the drop-down to share it with people who have access to the link. You could also share it with people who do not have access by selecting **Anyone with the link** from the drop-down.

Oracle Autonomous Database Privacy Policy Details

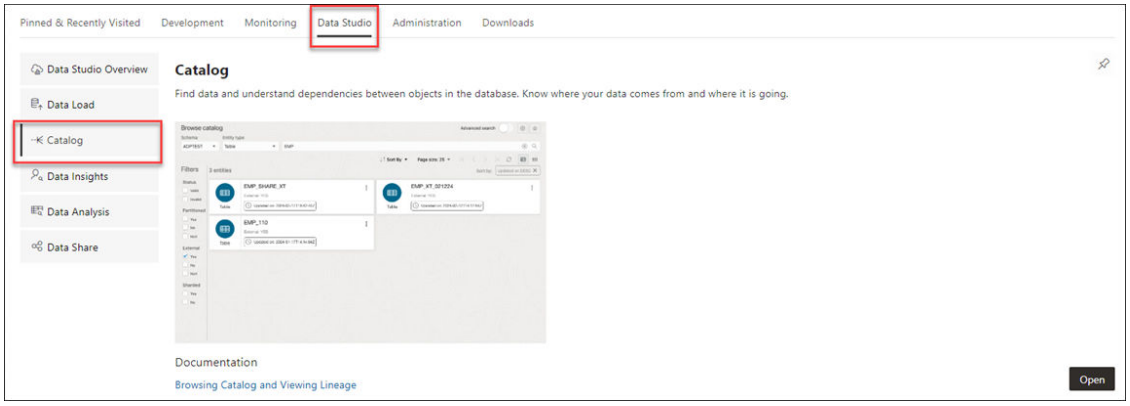
This topic covers details for writing policies to control access to Autonomous Database resources.

The **Oracle Autonomous Database Add-on for Google Sheets** must comply with Privacy Policy. The **Oracle Autonomous Database add-on for Google Sheet's** use and transfer to any other app of information received from Google APIs will adhere to Google API Services User Data Policy, including the Limited Use requirements.

The Catalog Page

The Catalog page displays information about entities in the Oracle Autonomous Database.

To reach the Catalog page, select the **Catalog** menu in the **Data Studio** tab of the Launchpad.



or click the Selector icon and select **Catalog** from the Data Studio menu in the navigation pane.

> **Note:**
>
> If you do not see the Catalog card then your database user is missing the required DWROLE role.

The following topics describe the Catalog page and how to use it.

- About the Catalog Page
- Sorting and Filtering Entities
- Searching for Entities
- Viewing Entity Details

- About the Catalog Page
  Use the Catalog page to get information about the entities in and available to your Oracle Autonomous Database. You can see the data in an entity, the sources of that data, the objects that are derived from the entity, and the impact on derived objects from changes in the sources.
- Registering Cloud Links to access data
  Cloud Links enable you to remotely access read only data on an Autonomous Database instance.
- Export Data to Cloud
  Use the **Export Data to Cloud** menu in the table actions of the Browse Catalog page to export data as text from an Autonomous Database to a cloud Object Store. The text format export options are CSV, JSON, Parquet, or XML.
- Sorting and Filtering Entities
  On the Catalog page, you can sort and filter the displayed entities.
- Searching for Entities
  Search for entities in the Catalog by entering a search string (query) into the **Search catalog** field at the top of the Catalog page. You can type or paste in a new search string, edit an existing one, or construct a search string by clicking in the field and selecting items from the drop-down list.
- Gathering Statistics for Tables
  You can generate statistics that measure the data distribution and storage characteristics of tables.
- Viewing Entity Details

  To view details about an entity, click the **Actions** icon at the right of the entity entry, then click **View Details**.
- Editing Tables
  You can create and edit objects using Edit Table wizard available from the **Edit** menu in **Actions** (three vertical dots) besides the table entity.

Registering Cloud Links to access data

Cloud Links enable you to remotely access read only data on an Autonomous Database instance.

You can register a table for remote access for a selected audience. Scope indicates who can remotely access the data. Scope can be set to various levels, including to the region where the database resides, to individual tenancies, or to compartments. You can provide a namespace and a name other than the original schema and object names. For example, you can register a table under the namespace FOREST and for security purposes or for naming convenience, you can provide a namespace and a name other than the original schema and object names.

To register a table to cloud link, click Actions next to the Table entity and select **Register to Cloud Link**.

Specify the following field values:

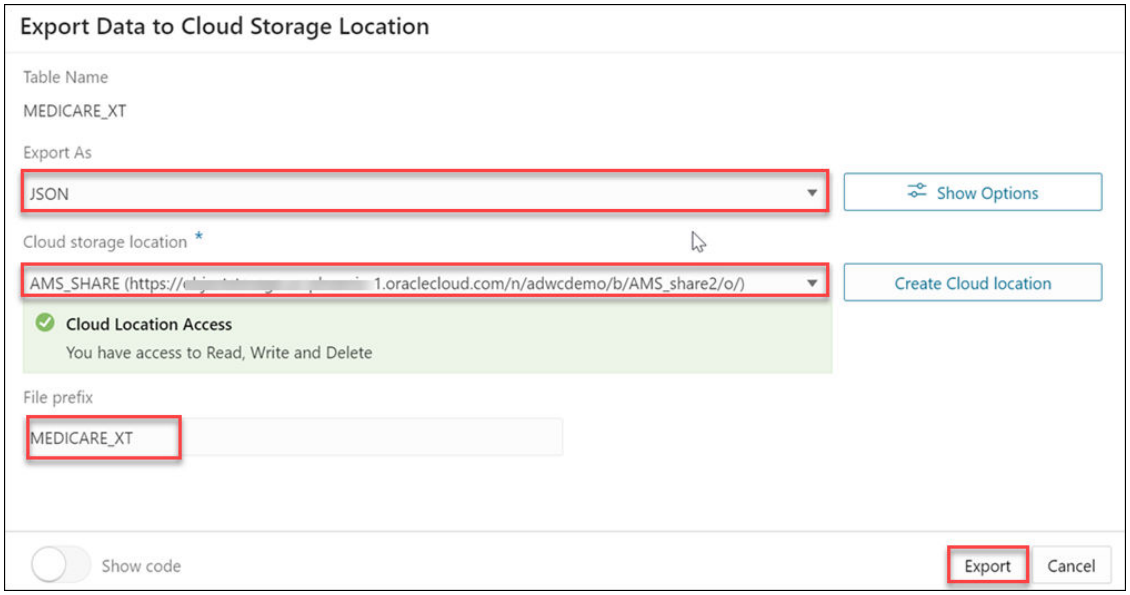
- **Cloud Link Namespace**: Enter a namespace to register the table. For example, ADMIN.
- **Cloud Link Name**: Enter a name for the cloud link. For example, Sales.
- **Description**: Specify the description of your cloud link.
- **Scope**: Specifies who and from where a user is allowed to access the registered table. You can choose from any of the following available options:
  - **MY$REGION**: You can grant remote data access to other tenancies in the region of the Autonomous Database instance that is registering the data set. This is the least restrictive scope.
  - **MY$TENANCY**: You can grant remote data access to any resource, tenancy, compartment, or database in the tenancy of the Autonomous Database instance that is registering the data set. This scope is more restrictive than MY$REGION scope.
  - **MY$COMPARTMENT**: You can grant remote data access to any resource, compartment, or database in the compartment of the Autonomous Database instance that is registering the data set.
  - **OCID**: Access to the data set is allowed for the specific Autonomous Database instances identified by OCID.

Click **OK** to finish the registration of the selected table with the cloud link.

Export Data to Cloud

Use the **Export Data to Cloud** menu in the table actions of the Browse Catalog page to export data as text from an Autonomous Database to a cloud Object Store. The text format export options are CSV, JSON, Parquet, or XML.

To export data to a cloud storage location, click **Actions** next to the Table entity on the Browse Catalog page and select **Export Data to Cloud**.



The Export Data to Cloud Storage Location displays the following fields:

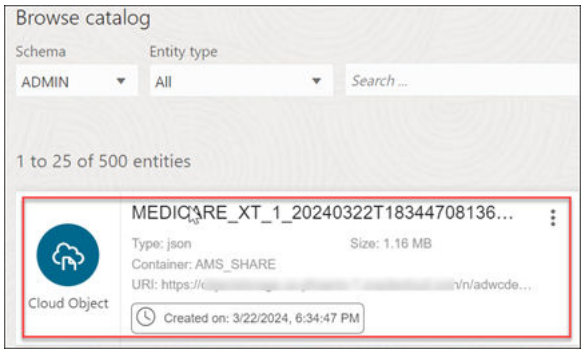
- **Table Name**: This field displays the table name.
- **Export As**: This drop-down displays the list of format options you wish to export the data. You can select CSV, JSON, Parquet, or XML. When you export the data as **CSV**, and click **Show Options**, you can:
  - Set the maximum file size value using the slider. The maximum file size you can export is from 10MB to 1GB.
  - Select the **header** option to write column names as the first line in output files.
  - Select the **Escape** option to specify the occurrence of the quote character in the field value using the "\" character.
  - **Text Enclosure and Field delimiter**: These options are visible only when the selected file is in CSV format. Select or enter the character used in the source file for text enclosure and field delimiter.
  - **Compression**: Select **No Compression** to disable compression or **GZIP** to enable GZIP compression for selected file.
- **Cloud storage Location**: Select the uri value to the URL for an existing bucket on your Cloud Object Storage from the drop-down. If you do not have a cloud storage location, select **Create Cloud Location** to create a cloud storage location to export your data. For more details, see Create Credentials.

> **Note:**
>
> The Cloud storage location is highlighted in red if the tool doesn't accept the selected cloud location. In this case, you will receive a Cloud Location Access Error that says "`You do not have access to Read, Write and Delete`".

- **File prefix**: Specify the File prefix. This will be the prefix of the file you export.

Click **Export**.

After the file export is complete, you will view the newly created cloud object in the entity list of the Browse Catalog page.



Gathering Statistics for Tables

You can generate statistics that measure the data distribution and storage characteristics of tables.

You must gather statistics periodically for tables where the statistics become stale over time because of changing data volumes or changes in column values. The Gather Statistics icon gathers new statistics after the table's structure are modified in ways that make the previous statistics inaccurate. For example, after loading a significant number of rows into a table, collect new statistics on the number of rows. After updating data in a table, you do not need to collect new statistics on the number of rows, but you might need new statistics on the average row length.

Table Statistics also include column statistics. The statistics you generate include the following:

Table statistics:

- **Table Size**: Specifies the size of the table in bytes.
- **Number of rows**: Displays the number of rows in the table.
- **Number of columns**: Displays the number of columns in the table
- **Compressed size**: Displays the size of compressed table in bytes.
- **Statistics gathered on**: Displays date and time of last statistics gathered.

Column Statistics

- Number of distinct values (NDV) in column
- Number of nulls in column
- Data distribution (histogram)

The above information is displayed in the statistics details of tables.

Editing Tables

You can create and edit objects using Edit Table wizard available from the **Edit** menu in **Actions** (three vertical dots) besides the table entity.

Clicking **Edit** from the Actions menu opens the **Edit Table** wizard. You can visit the panes in any order to edit a table. The table properties are grouped in several panes.

- **Schema**: Database schema in which the table exists.

- **Name**: Name of the table.

The different panes in the dialog are described in the following sections:

- Columns Pane
- Primary Key Pane
- Unique Keys Pane
- Indexes Pane
- Foreign Keys Pane
- Table Constraints Pane
- Comments Pane
- Storage Pane
- External Table Properties Pane
- Materialized View Pane
- DDL Pane
- Output Pane

- Columns Pane
  Specifies properties for each column in the table.
- Primary Key Pane
  Specifies the primary key for the table.
- Unique Keys Pane
  Specifies one or more unique constraints for the table.
- Indexes Pane
  Lists the indexes defined for the table.
- Foreign Keys Pane
  Specifies one or more foreign keys for the table.
- Table Constraints Pane
  Specifies one or more check constraints for the table.
- Comments Pane
  Enter descriptive comments in this pane.
- Storage Pane
  Enables you to specify storage options for the table.
- External Table Properties Pane
  Specifies options for an external table.
- Materialized View Pane
  Specifies options for a materialized view.
- DDL Pane
  You can review and save the SQL statements that are generated when creating or editing the object. If you want to make any changes, go back to the relevant panes and make the changes there.
- Output Pane
  Displays the results of the DDL commands.

Columns Pane

Specifies properties for each column in the table.

General tab

Lists the columns available in the table.To add a column, click **Add Column (+)**. A new row is added to the table below. Select the row and enter the details for the column. To delete a column, select the row and click **Remove Column (-)**. To move a column up or down in the table, select it and use the up-arrow and down-arrow icons.

- **Name**: Name for the column.
- **Datatype**: Data type for the column.
- **Default**: If no value is specified, the default value is null.
- **Default on NULL**: Applicable for Oracle Database 12c and later releases. If this option is selected, when a row is inserted into the table and the value specified for the column is NULL, the default value is inserted into the column.
- **Expression**: Expression for computing the value in the column.
- **Comments**: Optional descriptive comments about the column. Use this field to provide descriptions for the attributes.

In the table:

- **PK**: If this option is selected, the column becomes the primary key.
- **Identity Column** : If this option is selected, the column becomes an identity column. This is applicable only for Oracle Database 12c and later releases. For more details, see the Identity Column tab.

Constraints tab

Displays the Not Null and Check Constraints for a column. A check constraint requires values in a column to comply with a specified condition.

- **Not Null Constraint: Name**: Name for the Not Null constraint.
- **Not Null Constraint: Not Null**: If this option is selected, the column must contain data. You cannot specify no value or an explicit null value for this column when you insert a row. If this option is not checked, the column can contain either data or no data. A primary key column cannot be null.
- **Check Constraint: Name**: Name for the check constraint definition.
- **Check Constraint: Constraint**: Condition that must be met for a column to fulfill the check constraint. You can use any valid CHECK clause (without the CHECK keyword). For example, to indicate that the value in a numeric column named RATING must be from 1 to 10, you can specify: rating >=1 and rating <= 10.
- **Enabled**: If this option is selected, the constraint is checked when data is entered or updated in the column.
- **Deferrable**: If this option is selected, you can defer checking the validity of the constraint until the end of a transaction.
- **Initially Immediate**: If this option is selected, the constraint is checked whenever you add, update, or delete data from the column.
- **Validate**: If this option is selected, the existing data is checked to see if it conforms to the constraint.

Primary Key Pane

Specifies the primary key for the table.

The primary key is the column, or set of columns, that uniquely identifies each row in the table. If the Primary Key checkbox is selected for a column in the General tab, the corresponding fields are automatically populated in the Primary Key pane. You can make changes to the properties as required.

An index is automatically created on the primary key.

- **Name**: Name of the constraint to be associated with the primary key definition.
- **Enabled**: If this option is checked, the primary key constraint is enforced: that is, the data in the primary key column (or set of columns) must be unique and not null.
- **Index**: Name of the index to which the primary key refers.Tablespace: Name of the tablespace associated with the index.
- **Available Columns**: Lists the columns that are available to be added to the primary key definition. You can select multiple attributes, if required, for the primary key.
- **Selected Columns**: Lists the columns that are included in the primary key definition. To add a column to the primary key definition, select it in Available Columns and click the Add (>) icon; to remove a column from the primary key definition, select it in Selected Columns and click the Remove (<) icon. To move all columns from available to selected (or the reverse), use the Add All (>>) or Remove All (<<) icon. To move a column up or down in the primary key definition, select it in Selected Columns and use the arrow buttons.

Unique Keys Pane

Specifies one or more unique constraints for the table.

A unique constraint specifies a column, or set of columns, whose data values must be unique: each data value must not be null, and it must not be the same as any other value in the column.

To add a unique constraint, click the Add button; to delete a unique constraint, select it and click the Remove button.

- **Name**: Name of the unique constraint.
- **Enabled**: If this option is selected, the unique constraint is enforced.
- **Rely**: If this option is selected, the constraint in NOVALIDATE mode is taken into account during query rewrite.
- **Deferrable**: If this option is selected, in subsequent transactions, constraint checking can be deferred until the end of the transaction using the SET CONSTRAINT(S) statement.
- **Initially Immediate**: If this option is selected, the constraint is checked at the end of each subsequent SQL statement.
- **Validate**: If the option is selected, the existing data is checked to see if it conforms to the constraint.
- **Index**: Name of the index to which the unique key refers.
- **Tablespace**: Name of the tablespace associated with the index.
- **Available Columns**: Lists the columns that are available to be added to the unique constraint definition.
- **Selected Columns**: Lists the columns that are included in the unique constraint definition.

To add a column to the unique constraint definition, select it in Available Columns and click the Add (>) icon; to remove a column from the unique constraint definition, select it in Selected Columns and click the Remove (<) icon. To move all columns from available to selected (or the reverse), use the Add All (>>) or Remove All (<<) icon. To move a column up or down in the unique constraint definition, select it in Selected Columns and use the arrow buttons.

Indexes Pane

Lists the indexes defined for the table.

To add an index, click Add Index (+); to delete an index, select it and click Remove Index (-).

- **Name**: Name of the index.
- **Type**: The type of Oracle index. Non-unique means that the index can contain multiple identical values; Unique means that no duplicate values are permitted; Bitmap stores rowids associated with a key value as a bitmap.
- **Tablespace**: Name of the tablespace for the index.
- **Expression**: A column expression is an expression built from columns, constants, SQL functions, and user-defined functions. When you specify a column expression, you create a function-based index.
- **Available Columns** and **Selected Columns**: Columns selected for the index. To select a column, click the column in the Available Columns box, and then click the click the Add Selected Columns icon to move it to the Selected Columns box.

Foreign Keys Pane

Specifies one or more foreign keys for the table.

A foreign key specifies a column ("local column"), whose data values match values in the primary key or unique constraint of another table.

- **Name**: Name of the foreign key definition.
- **Enabled**: If this option is checked, the foreign key is enforced.
- **Rely**, **Deferrable**, **Initially Immediate**, **Validate**: See the description of these fields in the Unique Keys pane.
- **Referenced Constraint: Schema**: Name of the schema containing the table with the primary key or unique constraint to which this foreign key refers.
- **Referenced Constraint: Table**: Name of the table with the primary key or unique constraint to which this foreign key refers.
- **Referenced Constraint: Constraint**: Name of the primary key or unique constraint to which this foreign key refers.
- **Referenced Constraint: On Delete**: Action to take automatically when a row in the referenced table is deleted and rows with that value exist in the table containing this foreign key: NO ACTION (shown by a crossing line in diagrams) performs no action on these rows; CASCADE (shown by an "X") deletes these rows; SET NULL (shown by a small circle) sets null all columns in those rows that can be set to a null value.
- **Associations: Local Column**: Lists the column in the currently selected (local) table that is included in the foreign key definition. For each referenced column in the foreign key definition, select the name of a column in the edited table.

- **Associations: Referenced Column**: For each local column, identifies the column in the other (foreign) table that must have a value matching the value in the local column.

Table Constraints Pane

Specifies one or more check constraints for the table.

A check constraint specifies a condition that must be met when a row is inserted into the table or when an existing row is modified.

- **Name**: Name of the check constraint definition.
- **Check Condition**: Condition that must be met for a row to fulfil the check constraint. You can use any valid CHECK clause (without the CHECK keyword). For example, to indicate that the value in a numeric column named RATING must be from 1 to 10, you can specify rating >=1 and rating <= 10.
- **Enabled**: If this option is checked, the check constraint is enforced.

Comments Pane

Enter descriptive comments in this pane.

This is optional.

Storage Pane

Enables you to specify storage options for the table.

When you create or edit a table or an index, you can override the default storage options.

- **Organization**: Specifies that the table is stored and organized with (Index) or without an index (Heap) or as an external table (External).
- **Tablespace**: Name of the tablespace for the table or index.
- **Logging**: ON means that the table creation and any subsequent INSERT operations against the table are logged in the redo log file. OFF means that these operations are not logged in the redo log file.
- **Row Archival**: YES enables in-database archiving, which allows you to archive rows within the table by marking them as invisible.

External Table Properties Pane

Specifies options for an external table.

An external table is a read-only table whose metadata is stored in the database but whose data in stored outside the database.

External Table

- **Access Driver Type**: Specifies the type of external table.
  - **ORACLE_LOADER**: Extracts data from text data files. This is the default access driver, which loads data from external tables to internal tables.
  - **– ORACLE_DATAPUMP**: Extracts data from binary dump files. This access driver can perform both loads and unloads.
  - **– ORACLE_BIGDATA**: Extracts data from Oracle Big Data Appliance.
  - **– ORACLE_HDFS**: Extracts data stored in a Hadoop Distributed File System (HDFS).

  - – **ORACLE_HIVE**: Extracts data stored in Apache HIVE.

- **Default Directory**: Specifies the default directory to use for all input and output files that do not explicitly name a directory object. The location is specified with a directory object, not a directory path.
- **Access Params**: Assigns values to the parameters of the specific access driver for the external table. Access parameters are optional.
  - **OPAQUE_FORMAT_SPEC**: The opaque_format_spec specifies all access parameters for the ORACLE_LOADER, ORACLE_DATAPUMP, ORACLE_HDFS, and ORACLE_HIVE access drivers. For descriptions of the access parameters, see *Oracle Database Utilities*. Field names specified in the opaque_format_spec must match columns in the table definition, else Oracle Database ignores them.
  - **USING CLOB**: Enables you to derive the parameters and their values through a subquery. The subquery cannot contain any set operators or an ORDER BY clause. It must return one row containing a single item of data type CLOB.
- **Reject Limit**: The number of conversion errors that can occur during a query of the external data before an Oracle Database error is returned and the query is aborted.
- **Project Column**: Determines how the access driver validates the rows of an external table in subsequent queries.
  - **ALL**: Processes all column values, regardless of which columns are selected, and validates only those rows with fully valid column entries. If any column value raises an error, such as a data type conversion error, the row is rejected even if that column was not referenced in the select list of the query.
  - **REFERENCED**: Processes only those columns in the select list of the query. The ALL setting guarantees consistent result sets. The REFERENCED setting can result in different numbers of rows returned, depending on the columns referenced in subsequent queries, but is faster than the ALL setting. If a subsequent query selects all columns of the external table, then the settings behave identically.
- **Location**: Specifies the data files for the external table. Use the Add (+) icon to add each location specification.
  - ORACLE_LOADER and ORACLE_DATAPUMP, the files are named in the form directory:file. The directory portion is optional. If it is missing, then the default directory is used as the directory for the file. If you are using the ORACLE_LOADER access driver, then you can use wildcards in the file name. An asterisk (*) signifies multiple characters and a question mark (?) signifies a single character.
  - For ORACLE_HDFS, LOCATION is a list of Uniform Resource Identifiers (URIs) for a directory or for a file. There is no directory object associated with a URI.
  - For ORACLE_HIVE, LOCATION is not used. Instead, the Hadoop HCatalog table is read to obtain information about the location of the data source (which could be a file or another database).

Opaque Format Spec

Specifies all access parameters for the ORACLE_LOADER, ORACLE_DATAPUMP, ORACLE_HDFS, and ORACLE_HIVE access drivers.

CLOB Subquery

Type or copy and paste the query.

Materialized View Pane

Specifies options for a materialized view.

Query: Contains the SQL code for the query part of the view definition. Type or copy and paste the query.

General

- **On Pre-built Table**: If **Yes**, an existing table is registered as a preinitialized materialized view. This option is particularly useful for registering large materialized views in a data warehousing environment. The table must have the same name and be in the same schema as the resulting materialized view, and the table should reflect the materialization of a subquery.
- **Reduced Precision**: **Yes** authorizes the loss of precision that will result if the precision of the table or materialized view columns do not exactly match the precision returned by the subquery. If **No**, the precision of the table or materialized view columns must exactly match the precision returned by the subquery, or the create operation will fail.
- **For Update**: Select **Yes** to allow a subquery, primary key, object, or rowid materialized view to be updated. When used in conjunction with Advanced Replication, these updates will be propagated to the master.
- **Real Time MV**: Select **Yes** to create a real-time materialized view or a regular view. A real-time materialized view provides fresh data to user queries even when the materialized view is not in sync with its base tables due to data changes. Instead of modifying the materialized view, the optimizer writes a query that combines the existing rows in the materialized view with changes recorded in log files (either materialized view logs or the direct loader logs). This is called on-query computation.
- **Query Rewrite**: If **Enable**, the materialized view is enabled for query rewrite, which transforms a user request written in terms of master tables into a semantically equivalent request that includes one or more materialized views.
- **Build**: Specifies when to populate the materialized view. **Immediate**indicates that the materialized view is to be populated immediately. **Deferred** indicates that the materialized view is to be populated by the next refresh operation. If you specify Deferred, the first (deferred) refresh must always be a complete refresh; until then, the materialized view has a staleness value of unusable, so it cannot be used for query rewrite.
- **Use Index**: If **Yes**, a default index is created and used to speed up incremental (fast) refresh of the materialized view. If **No**, this default index is not created. (For example, you might choose to suppress the index creation now and to create such an index explicitly later.)
- **Index Tablespace**: Specifies the tablespace in which the materialized view is to be created. If a tablespace is not selected, the materialized view is created in the default tablespace of the schema containing the materialized view.
- **Cache**: If **Yes**, the blocks retrieved for this table are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed. This setting is useful for small lookup tables. If **No**, the blocks are placed at the least recently used end of the LRU list.

Refresh Clause

- **Refresh**: Select **Yes** to enable refresh operations.
- **Refresh Type**: The method of refresh operation to be performed:

  - Complete Refresh: Executes the defining query of the materialized view, even if a fast refresh is possible.
  - Fast Refresh: Uses the incremental refresh method, which performs the refresh according to the changes that have occurred to the master tables. The changes for conventional DML changes are stored in the materialized view log associated with the master table. The changes for direct-path INSERT operations are stored in the direct loader log.
  - Force Refresh: Performs a fast refresh if one is possible; otherwise, performs a complete refresh.
- **Action**: The type of refresh operation to be performed:
  - On Demand: Performs a refresh when one of the DBMS_MVIEW refresh procedures are called.
  - On Commit: Performs a fast refresh whenever the database commits a transaction that operates on a master table of the materialized view. This may increase the time taken to complete the commit, because the database performs the refresh operation as part of the commit process.
  - Specify: Performs refresh operations according to what you specify in the Start on and Next fields.
- **Start Date**: Starting date and time for the first automatic refresh operation. Must be in the future.
- **Next Date**: Time for the next automatic refresh operation. The interval between the Start on and Next times establishes the interval for subsequent automatic refresh operations. If you do not specify a value, the refresh operation is performed only once at the time specified for Start on.
- **With**: Refresh type, which determines the type of materialized view:
  - Primary Key: Creates a primary key materialized view, which allows materialized view master tables to be reorganized without affecting the eligibility of the materialized view for fast refresh.
  - Row ID: Creates a rowid materialized view, which is useful if the materialized view does not include all primary key columns of the master tables.
- **Default Storage**: If Yes, DEFAULT specifies that Oracle Database will choose automatically which rollback segment to use. If you specify DEFAULT, you cannot specify the `rollback_segment`. DEFAULT is most useful when modifying, rather than creating, a materialized view.
- **Storage Type**: MASTER specifies the remote rollback segment to be used at the remote master site for the individual materialized view. LOCAL specifies the remote rollback segment to be used for the local refresh group that contains the materialized view. This is the default.
- **Rollback Segment**: Enter the name of the rollback segment.
- **Using Constraint**: If this option is checked, more rewrite alternatives can be used during the refresh operation, resulting in more efficient refresh execution. The behavior of this option is affected by whether you select Enforced or Trusted.
  - Enforced: Causes only enforced constraints to be used during the refresh operation.
  - Trusted: Enables the use of dimension and constraint information that has been declared trustworthy by the database administrator but that has not been validated by the database. If the dimension and constraint information is valid, performance may

improve. However, if this information is invalid, then the refresh procedure may corrupt the materialized view even though it returns a success status.

DDL Pane

You can review and save the SQL statements that are generated when creating or editing the object. If you want to make any changes, go back to the relevant panes and make the changes there.

For a new table, click **CREATE** to view the generated DDL statements.

When you edit table properties, click **UPDATE** to view the generated ALTER statements.

For a new table, the UPDATE tab will not be available. When you are finished, click **Apply**.

Output Pane

Displays the results of the DDL commands.

If there are any errors, go to the appropriate pane, fix the errors, and run the commands again. You can save to a text file or clear the output.

About the Catalog Page

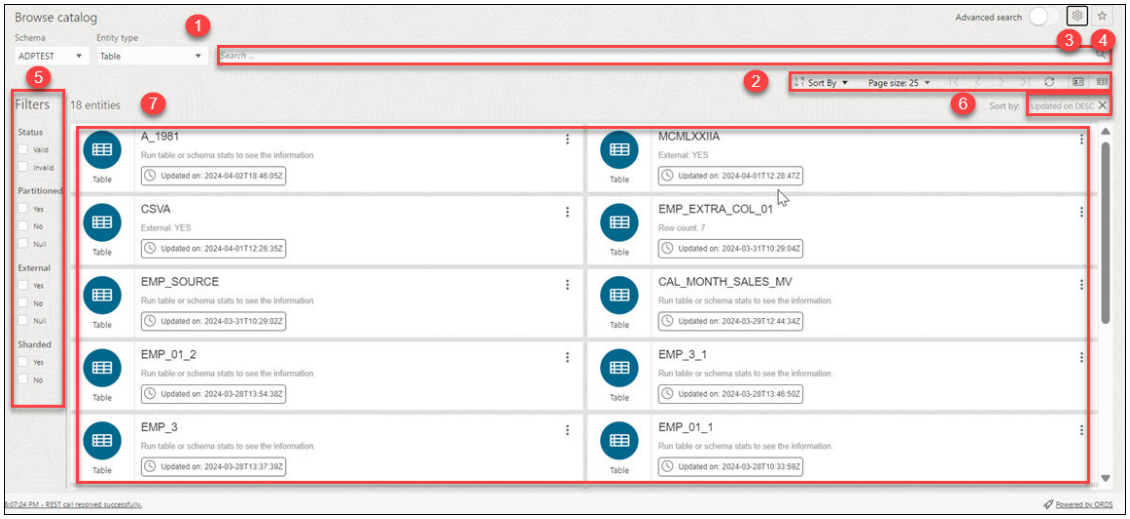
Use the Catalog page to get information about the entities in and available to your Oracle Autonomous Database. You can see the data in an entity, the sources of that data, the objects that are derived from the entity, and the impact on derived objects from changes in the sources.

The Catalog page lists and displays details about:

- Database objects such as business models, cloud storage links, and tables that are created by Database Actions applications such as Data Analysis and Data Load.
- Database data dictionary objects such as tables, columns, database links, analytic views, packages, and procedures that have been created by the database tools or a database application such as SQL Developer.

When you first open the Browse catalog page, it contains the **Search catalog** field, the **Show User Preferences**  button, a list of recently viewed objects, and several suggested searches. You can enter a search string, click on one of the recent objects to see its details, or click on **Show search suggestions** icon to view suggestions on the right side of the page.

As soon as you select Catalog menu from Data Studio menu bar, the page is displayed as below.



The Catalog page contains:

1. **Search catalog** field

   You can search for the entities by two ways.

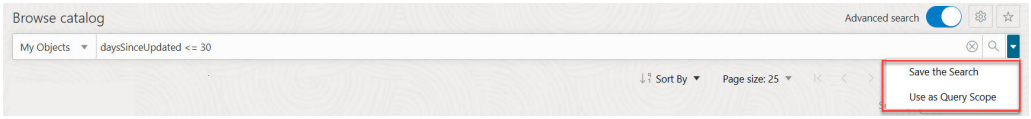
   - **Simple search:** By default you will be able to search with Simple search. With simple search:

     a. Select the schema from the Schema drop-down from where you would like to browse the entity. The drop-down list is sorted alphabetically.

     b. Select the Entity type from the drop-down. For example, Table. The drop-down list is sorted alphabetically.

     c. Click the search field and type or paste a string which consists of the name of the entity you wish to view. You can also click the field and edit a search string already present in the field.

   - **Advanced Search:** Turn on the **Advanced Search** toggle on top of the search catalog field to search for an entity by constructing a search query based on type of entity, schema, property of the entity, etc. From the browse catalog objects drop-down list, select the object from where you want to search your entities. Click the search field and build a query by selecting items from the list of suggestions displayed as a drop-down list. You can modify an existing query or build a new one.

     After you have finished constructing the search query, select **Save the Search** from the drop-down next to the magnifier icon to save the save the current search query.

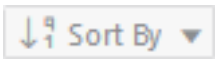
     You can also select **Use as Query Scope** from the drop-down in the end of the search field to take the current search query and turn it to saved query scope.

   Click **X** to clear your searches in the Catalog field.

   For details about how to construct a search string, see Searching for Entities.

2. **Toolbar**

   The toolbar appears after you run an initial search. It contains these buttons:

   - **Sort By** 

To select sorting values, click the **Sort By** button to open the list of options. Then click the **Ascending** or **Descending** icon next to one or more of the sorting values. For example, if you select the **Ascending** icon next to **Entity name** and the **Descending** icon next to **Entity type**, the entities will be sorted in alphabetical order by entity name and then in reverse alphabetical order by entity type.

Click **Reset** in the list to clear the choices in the list.

The sorting values you choose are listed next to the **Sort by** label beneath the toolbar. Click the **X** icon on a sorting value to remove it.

- **Page size** 

  By default, up to 25 entities are displayed on the page. If you want more entities on a page, select a number from this list.

- **Previous** and **Next**

  If the search results are displayed on multiple pages, click these buttons to navigate through the pages.

- **Refresh** 

  Click to refresh the entities shown on the page, based on the current search string.

- **Entity view options** 

  Choose one of these three options to set how entities are displayed on the page.

  Click **Open Card view**  to display entities as card arranged into one or two columns; click **Open Grid View**  to display entities as rows in a table; or click **Open List View**  to display entities in a single column of panels.

  The above views display information about the entity, including name, type, owner, application, and other details, depending on the entity view option and on the entity type.

  Click **Action** and select **View Details** to view the details of an entity. In the Card view and the List view, you can also click the name of the entity to show its details.

- **Saved Search Suggestions**

  Click to open or close the **Saved Search Suggestions** panel.

3. **Show User Preferences** 

   Click the **start** icon to show **Saved Searches suggestions**.

   When you first open the Catalog, this button appears to the top of the **Search catalog** field.

   Click **Show User Preferences**  to set the behavior of the Catalog. When you set these options, they take place immediately and are also saved as the default behavior for the page. Clicking Show User Preferences opens the General tab where you can view the following options.

   - **Show system tables**

Select this option to include system tables in the search results.

- **Show private tables**

  Select this option to include private tables in the search results.

- **Page size**

  Select the number of entities to display on the page.

- **Entities view**

  Select how entities are shown on the page. There are three options: **Card view** (default) displays entities as cards arranged into one or two columns; **Grid view** displays entities as rows in a table; and **List view** displays entities in a single column of panels.

- **Save last 5 search queries**

  Select this option to save the five previous search queries that you entered into the **Search catalog** field. When the **Search catalog** field is empty and you click in the field, the last five search queries are listed in the drop-down list. (Any predefined searches selected from the **Suggestions** panel won't appear in this list, unless you edited them to make a new search.)

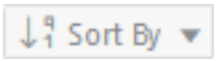
  Gradually progress to use the **Query Scopes** tab to view, create, and delete query scopes. You can search for catalogs and save this search using query scopes. See Searching for Entities for exploring this tab.

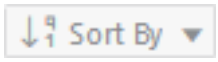
  Click on the **Saved Searches** tab to save the current search query as per the criteria you specify or create a new saved search.

4. **Filters** panel

   Select one or more filter values to limit the entities shown on the page. Only those entities that match the filter values are shown. That is, the items returned by a search are filtered by these filter settings. Selecting all or none of the options shows all entities. See Sorting and Filtering Entities.

5. **Sort by** settings

   When you set sorting values by using the **Sort By** 

control in the toolbar (see above), the settings are displayed in small boxes beneath the toolbar. You can delete a setting by clicking the **X** icon in the box. Or you can change the settings by returning to the **Sort By** 

control in the toolbar.

6. **Display area**

   The area beneath the **Search catalog** field displays the entities returned by a search and that match the filter criteria set in the **Filters** panel. You can sort the entities by clicking the **Sort By** button and then setting sort values. See Sorting and Filtering Entities.

   Click the name of the entity or click **Action** to view details about the entity. See Viewing Entity Details.

7. **Saved Searches Suggestions** panel

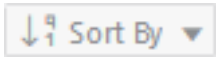
   The items listed under **Saved Searches Suggestions** represent predefined queries (search strings) that search for the items described in the link. For example, **All tables** is associated with the search string `type:TABLE`. When you select one of the suggested search strings, the string is inserted into the **Search catalog** field, the query is run, and the results are displayed on the page. You can modify the search string in the **Search catalog** field to modify the search.

Sorting and Filtering Entities

On the Catalog page, you can sort and filter the displayed entities.

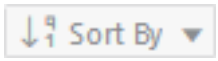
Sort Entities

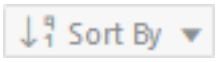
To set a sort order for the entities on the page,

1. Click **Sort by**  on the toolbar.
2. Find the value you want to use to sort, for example **Entity Type**.
3. Click the **Ascending**  icon or the **Descending**  icon next to that value.

   After you've selected a value, it is displayed in a box next to the **Sort by** label beneath the toolbar.
4. You can set one or more parameters. If you want to add a parameter to your search, repeat the steps above. For example, you can set **Entity name (ASC)** and then set **Created on (DESC)**. Both appear as separate boxes next to the **Sort by** label beneath the toolbar.

To remove a sort value, you can do either of the following:

- Click the **X** icon in the box that shows the sort value (next to the Sort by label beneath the toolbar).
- Click **Sort by**  on the toolbar and click the **Ascending** or **Descending**  icon next to the value you want to remove.

To return to the default (**Updated on DESC**), click **Sort by**  and select **Reset**.

Filter Entities

Restrict which entities that are returned by a search are displayed on the page by setting filters in the **Filters** panel on the left side of the Catalog page. Select one or more filter values. Only those entities that are returned by a search and that match the filter values are shown. Selecting all or none of the options shows all entities returned by the search.

By default, system tables and private tables are not displayed. To display them, click **Catalog User Preferences**  in the toolbar and then click the **Show system tables** slider or the **Show private tables** slider, or both.

Searching for Entities

Search for entities in the Catalog by entering a search string (query) into the **Search catalog** field at the top of the Catalog page. You can type or paste in a new search string, edit an existing one, or construct a search string by clicking in the field and selecting items from the drop-down list.

The search operates on the entities that meet the filter criteria you have set in **Filters** panel. To clear the **Search catalog** field, click the **X** icon.

Entering Search Strings

Syntax and instructions for creating search strings are presented below.

You can enter a search string into the **Search catalog** field in several ways:

- Click an item under **Suggestions** on the right side of the page. When you click one of these items, a search string is entered into the **Search catalog** field and the search is run. This is the default type of search named, Simple search.
- Type or paste a string directly. When you start typing, suggestions are shown in the drop-down list below the **Search catalog** field. You can keep typing or you can select a suggestion from the list (see next bullet).
- Select or construct a search string by clicking the **Search catalog** field and selecting items using **Advanced search** toggle button.

  If the **Search catalog** field is empty, the drop-down list contains basic search parameters you can select to start building the query. When you select one, or when you click one of the links under **Suggestions**, the string is inserted in the **Search catalog** field, and the drop-down list is updated to present possible additional parameters. You can choose the additional parameters, or edit the string directly, or both.

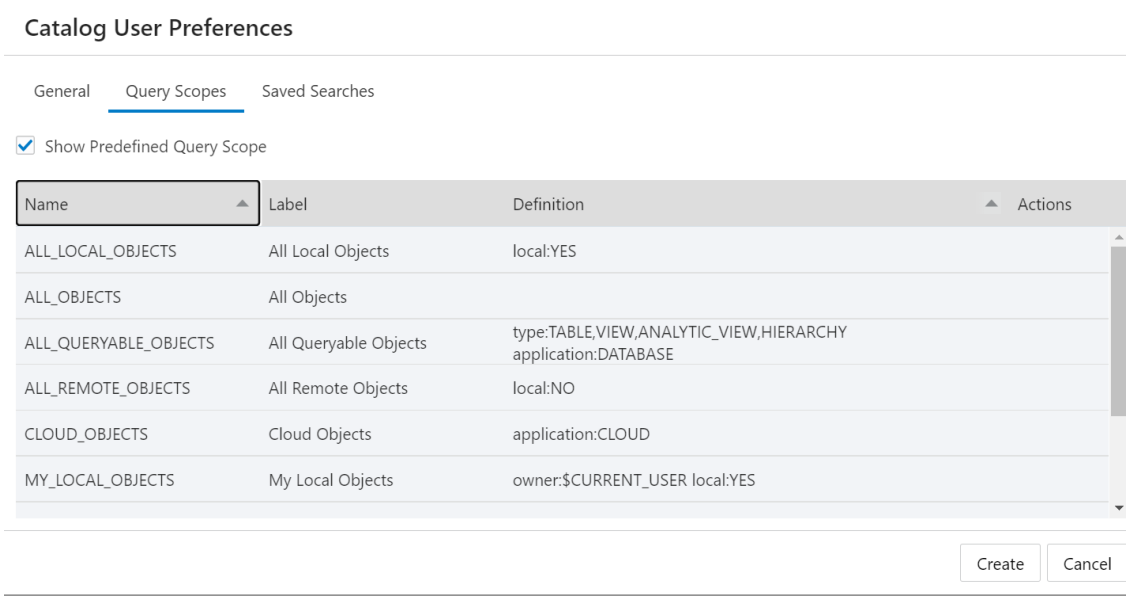
  If the **Search catalog** field is empty and if you set **Save last 5 search queries** in your preferences, the last five search strings that you entered into the **Search catalog** field will also appear in the drop-down. (Searches initiated by clicking one of the items under **Suggestions** won't appear in this list.)

When you finish constructing the string, press **Enter** or click the arrow button on the right side of the field.

Click the **magnifier** icon at the end of the Search field to accept the default query and search for all the current owner's tables, views, and analytic views.

Enter a string in the **Search Catalog** field to find the entities (schema, table, or view) whose label includes the specific string you enter. Click on the **Use as Query Scope** icon to save your search. This allows you to store frequently used searches and access them instantly without any inconvenience.

Selecting Query Scope icon prompts a **Catalog User Preferences** wizard which enables you to save your catalogs based on your preferences.



You can view, create, and edit the previously saved query scopes in this wizard. The query scopes are categorized based on by whom its created.

- Select **Custom** to view, create, edit, or delete the query scopes created by you.
- Select **Predefined** to view the query scopes which are already defined by the Database Actions. This option does not allow you to create, edit, or delete the query scopes.
- Select **All** to view all the query scopes. This includes Custom and Predefined query scopes.

Specify the following fields in the Query Scope tab.

- **Name**: Enter the name of the Query Scope. This is a mandatory field.
- **Label**: This is a mandatory field. Enter a descriptive name here. You will use this field to refer to a query scope.
- **Definition**: Enter the Oracle Autonomous Database Data Definition Language (DDL) that creates the search entity. This is the same search criteria you enter in the Search Catalog field.

Click **Create** to create the Query Scope. Click **Cancel** to cancel its creation.

Once you have created the new query scope, it is visible in the list of query scopes in the Catalog User Preferences wizard. Click **New** to create a new query scope

This wizard also appears on selecting **Catalog User Preferences**. See Catalog User Preferences for details.

Create a saved search

You can save your time from redefining the same search again in the future. You can diagnose problems faster since you are just few clicks away from accessing a saved search. Here is how you can create a saved search. Ensure the Advanced search toggle is on.

1. From the browse objects drop-down list, select the objects from where you want to search the entities (schema, table, or view).
2. Specify the search criteria in the **Search Catalog** field. For example, you want to search for a specific Entity type and a specific owner.

3. Select **Save the Search** from the drop-down next to the magnifier icon.
4. Clicking the icon prompts the Catalog User Preferences wizard.
5. Enter the name of the saved search in the Title field. The wizard automatically generates the Scope of the search The definition of the search is also automatically created by concatenating fields used in the **Search Catalog** field, for example, type: TABLE units AND owner: ADPTEST.
6. Enter description of the search in Description filed. This is not a mandatory step.
7. Select **Create** to create the saved search. Click **Cancel** to cancel its creation.

After the creation of the saved search, it appears on the list of Saved Searches.

You can change the columns displayed in the search results by clicking the pencil icon in the Actions column. Click the **delete** icon in the Actions column to delete the search you save. The saved searches you create are available for selection in the Saved Search panel in the right of the Catalog page.

Click **New** to create a new saved search.

This wizard also appears on selecting **Catalog User Preferences**. See Catalog User Preferences for details.

Basic Structure

The query string consists of a set of *search terms*. Here are some examples.

- `sales`
- `type:TABLE`
- `owner!=SH`
- `#deployment`
- `type:TABLE,VIEW`

Combine search terms by using the two Boolean operators `AND` and `OR`:

- `sales and type:TABLE`
- `sales or type:TABLE`

If you don't specify an explicit operator, then `AND` is assumed. All of the following are equivalent:

- `sales type:TABLE owner:sh`
- `sales AND type:TABLE owner:sh`
- `sales type:TABLE AND owner:sh`
- `sales AND type:TABLE AND owner:sh`

Negate a single search term by prefacing it by either `NOT` or by a - (hyphen). All of the following exclude tables from the search:

- `sales and NOT type:TABLE`
- `sales -type:TABLE`

Enclose search terms in parentheses to control order:

- `sales and (type:TABLE or type:VIEW)`
- `sales and NOT (type:TABLE or type:VIEW)`

If you don't add parentheses, the operators are read from left to right. The following are equivalent:

- `sales or type:TABLE and owner:SH`
- `(sales or type:TABLE) and owner:SH`

Search Terms

Search terms come in three forms:

- Simple Search Terms
- Property Search Terms
- Classification Search Terms

Simple Search Terms

A simple search term is a simple string, with or without quotes.

- Unquoted string: `sales`
- Single-quoted string: `'sales'`
- Double-quoted string: `"sales"`

All three types compare the given value, `sales` in the examples above, to the `ENTITY_NAME`, but they differ in how the comparison works, as shown in the table below:

Search String	Equivalent SQL WHERE Clause	Comment
`sales`	`WHERE REGEXP_LIKE(entity_name, 'SALES')`	A plain regular expression.
`'sales'`	`WHERE REGEXP_LIKE(entity_name, 'sales')`	Similar to the unquoted version, but the search term is not automatically converted to uppercase.
`sales costs`	`WHERE REGEXP_LIKE(entity_name, 'SALES') AND REGEXP_LIKE(entity_name, 'COSTS')`	Treated as two independent search terms combined with the `AND` operator.
`'sales costs'`	`WHERE REGEXP_LIKE(entity_name, 'sales costs')`	Treated as a single search term. You use single quotes to handle spaces and other special characters, such as colons.
`'don''t'`	`WHERE REGEXP_LIKE(entity_name, 'don''t')`	Escaped single quotes within a search string.
`"sales"`	`WHERE entity_name = 'sales'`	The search string is used as an exact match.

Property Search Terms

A *property search term* is a combination of three items:

1. The name of a search property (for example, `name`, `type`, or `owner`)
2. A search operator (for example, `:`, `>`, or `!=`)
3. A search value (for example, `sales`, `'sales'`, or `"sales"`)

For example:

- `name:sales`
- `owner=SH`
- `daysSinceCreated>20`

You can also specify a comma-delimited list of search values when using the operators `:`, `=`, or `~=`.

- If the operator is `:` or `=`, then the condition works like an `IN LIST`. For example, `name:X,Y` is equivalent to `entity_name IN ('X', 'Y')` and to `(entity_name = 'X' OR entity_name = 'Y')`.
- If the operator is `!=`, then the condition works like a `NOT IN LIST`. For example, `type!=TABLE,VIEW` is equivalent to `entity_type NOT IN ('TABLE', 'VIEW')` and to `(entity_type != 'TABLE' AND entity_type != 'VIEW')`.

The property name must be one of the following predefined strings. Property names are case sensitive, so `TYPE`, for example, is not supported.

Some properties apply to all entity types, and some properties apply only to a specific entity type, as shown in the two tables below.

The following table shows those properties that apply to all entity types.

Property	Meaning
`application`	The name of the entity application as defined by the `ALL_LINEAGE_APPLICATIONS` view. Examples include `DATABASE` and `INSIGHTS`.
`created`	The timestamp when the entity was created.
`dateCreated`	The date when the entity was created. This is the same as `created`, but truncated to the nearest day.
`dateUpdated`	The date when the entity was last updated. This is the same as `updated`, but truncated to the nearest day.
`daysSinceCreated`	The number of days since the entity was created. The value is zero if the entity was created today.
`daysSinceUpdated`	The number of days since the entity was last updated. The value is zero if the entity was updated today.
`link`	The name of the database link where the entity is defined. This can be used to search entities in other, linked, databases.
`local`	The value `YES` if the entity is defined within the database itself; otherwise, the value `NO`. Tables or Insights defined in a schema are examples of local entities. Objects in cloud storage, such as CSV or Parquet files, are examples of entities that are not local.
`name`	The name of the entity.
`namespace`	The namespace of the entity as defined by `ALL_LINEAGE_NAMESPACES`.
`oracleMaintained`	The value `YES`, if the entity is created and maintained by Oracle; otherwise, the value `NO`. The `ALL_TABLES` view is an example of an Oracle maintained entity.
`owner`	The owner of the entity.
`parent`	The full entity path of the parent entity, if it exists.
`parentName`	The name of the parent entity, if it exists.
`parentPath`	The full entity path of the parent entity, if it exists.

Property	Meaning
parentType	The type of the parent, if it exists.
path	The full path of the entity.
rootName	The name of the outermost containing entity. If the entity has no parent, then the rootName is equal to the entity name. If the entity does have a parent, then the rootName is defined, recursively, as the rootName of the parent entity.
rootNamespace	The namespace of the outermost containing entity. If the entity has no parent, then the rootNamespace is equal to the entity namespace. If the entity does have a parent, then the rootNamespace is defined, recursively, as the rootNamespace of the parent entity.
type	The entity type, as defined by ALL_LINEAGE_ENTITY_TYPES.
updated	The timestamp when the entity last updated.

Some searchable properties are specific to certain entity types. The following table shows those entity-type-specific properties. When searching for entities with these properties, it will speed up your search to specify the entity type. For example, instead of just searching on numRows, specify the entity type that has the numRows property:

```
TABLE AND numRows > 10
```

Entity Type	Properties
TABLE	numRows - The number of rows in the table. status - The status of the object: VALID or INVALID. partitioned - Indicates whether the table is partitioned (YES) or not (NO). external - Indicates whether the table is an external table (YES) or not (NO). sharded - Indicates whether the object is sharded (Y) or not (N).
COLUMN	dataType - The data type of the column. nullable - Indicates whether a column allows NULLs. The value is N if there is a NOT NULL constraint on the column or if the column is part of a PRIMARY KEY. Otherwise, the value is Y.
PACKAGE	status - The status of the object: VALID or INVALID.
PROCEDURE	status - The status of the object: VALID or INVALID.
ATTRIBUTE_DIMENSION	status - The status of the object: VALID or INVALID.
HIERARCHY	status - The status of the object: VALID or INVALID.
ANALYTIC_VIEW	status - The status of the object: VALID or INVALID.
MEASURE	dataType - Data type of the measure, such as NUMBER. nullable - Indicates whether a column allows NULLs. The value is N if there is a NOT NULL constraint on the column or if the column is part of a PRIMARY KEY. Otherwise, the value is Y. measureType - Type of the OLAP measure: • BASE - Base measures store the data • DERIVED - Derived measures calculate the data from base measures; also called calculated measures.
MINING_MODEL	status - The status of the object: VALID, INVALID, or N/A.

Entity Type	Properties
FUNCTION	`status` - The status of the object: `VALID`, `INVALID`, or `N/A`.
DIRECTORY	`status` - The status of the object: `VALID`, `INVALID`, or `N/A`.
EXTERNAL_LOCATION	`fileName` - The name of a file in Oracle Directories. `url` - The URI of file a file in object storage.
LIVE_TABLE_FEED	`enableNotifications` - Indicates whether a live table feed is enabled for notifications (`TRUE`) or not (`FALSE`).
ANALYTIC_DASHBOARD	`status` - The status of the object: `VALID` or `INVALID`.
ANALYTIC_REPORT	`status` - The status of the object: `VALID` or `INVALID`.
SHARE	`status` - The status of the object: `VALID` or `INVALID`.
SHARE_PROVIDER	`status` - The status of the object: `VALID` or `INVALID`.
SHARE_RECIPIENT	`status` - The status of the object: `VALID` or `INVALID`.

Note:

You can find the list of query-type-specific properties by running the following query on The SQL Page:

```
select *
from (
   select
      entity_type,
      JSON_QUERY(annotation, '$.searchProperties') properties
   from all_lineage_entity_types)
where properties is not null;
```

The operator must be one of the following.

Property	Meaning
`=`	Equal to
`!=`	Not equal to
`>`	Greater than
`>=`	Greater than or equal to
`<`	Less than
`<=`	Less than or equal to
`:`	Equal to For example, `type:TABLE` is the same as `type=TABLE`.
`~=`	Represents `REGEXP_LIKE` For example, the following two search items are equivalent: • `sales` • `name~=sales`

The value of the property search term is a string. As with the simple search term, this string can be unquoted, single-quoted, or double-quoted. Unquoted strings are converted to uppercase and quoted strings are left as they are.

Search String	Equivalent SQL WHERE Clause
`name=sales`	`WHERE entity_name = 'SALES'`
`name='sales'`	`WHERE entity_name = 'sales'`
`type:table`	`WHERE entity_type = 'TABLE'`
`parent~='COSTS'`	`WHERE REGEXP_LIKE(entity_parent_path, 'COSTS')`
`daysSinceCreated>10`	`WHERE TRUNC(CURRENT_TIMESTAMP AT TIME ZONE ''UTC'') - TRUNC("CREATED") > 10`
`owner!="DemoUser"`	`WHERE owner != 'DemoUser'`
`parentName:null`	`WHERE entity_parent_name IS NULL`
`parentName!=null`	`WHERE entity_parent_name IS NOT NULL`

Classification Search Terms

Classifications are metadata about an entity, such as a caption or a description. Classification search terms are similar to property search terms, but they work with entity classifications such as captions or descriptions instead of standard entity properties. The name of the classification must be prefixed by a hash tag (#) and may be unquoted or enclosed in double quotation marks ("). A query converts unquoted classifications to uppercase, which means that the following forms of the `CAPTION` classification are equivalent.

- `#caption`
- `#Caption`
- `#CAPTION`
- `#"CAPTION"`

Classifications are, by their nature, multi-lingual in the sense that the value can vary by National Language Support (NLS) language. By default, the query syntax uses the value of the classification in the current NLS language. You can specify a specific NLS language by adding the name of the language after a forward slash. The language can be quoted, but the query converts it to uppercase in all forms.

- `#caption/French`
- `#caption/"ITALIAN"`

If the name of the specified language contains a space (for example, `"CANADIAN FRENCH"`), you must either enclose the name in quotes or replace the space with an underscore character (or both). The following are equivalent:

- `#caption/Canadian_French`
- `#caption/"CANADIAN FRENCH"`
- `#caption/"Canadian_French"`

If you use a classification name on its own as a search term, then the search returns entities that have any non-null value for that classification. If you've defined a classification called `DEPLOYED`, for example, then you can see all entities with the classification by using a simple term:

- `#deployed`

As with other search terms, you can negate it using `NOT` or `-` (hyphen). To exclude all entities with the `DEPLOYED` classification, you can use either of the following searches:

- `NOT #DEPLOYED`
- `-#deployed`

You can also use classifications with operator/value pairs, with the same semantics used by property search items; for example:

- `#caption~=sales`
- `#description/Spanish=Ventas`

Searching on Multiple Properties

The default for this is `ENTITY_NAME`, so that if you enter a query string like "sales" it will search for entities with "sales" in the name.

You can also list multiple properties and classifications in this argument; for example, *`entity_name #caption #description`*, which will cause it to search for entities with `sales` in their name or in any `CAPTION` or `DESCRIPTION` classifications they may have.

Viewing Entity Details

To view details about an entity, click the **Actions** icon at the right of the entity entry, then click **View Details**.

For all entities, the details include Lineage and Impact sections. The inclusion of other details, such as Preview and Statistics, varies by entity type.

Preview

Preview displays the data of the entity. For a table, the Preview displays the columns of the table and the data in those columns. You can sort the data in the column into ascending or descending order by clicking the up or down arrow to the right of the column name.

Describe

For an analytic view, the Describe tab has information about the entity, and displays the hierarchies, levels, level depth, dimension tables, level columns, and number of distinct values for a level.

Lineage

Lineage displays all known information about the upstream dependencies of the entity, and therefore how the entity was created and how it is linked to other entities.

For example, for a table that you created in your database, the lineage is just the table. For a table that you created by loading a CSV file from cloud storage, the lineage includes the ingest directive for the data load and the CSV file that is the source of the data.

Pointing to the name of an item in the lineage displays the table name, the application that created it, the type of entity, the path to it, and the schema it is in.

Arrows point from an entity to the entity that it derives from. For example, for a table created in a data load job, an arrow points from the table to the ingest job and an arrow points from the ingest job to the CSV file. If you point to an arrow, then a Links Information box appears that shows information about the relationship between the two entities.

To view more details about an item, click the Actions icon for the item, then click **Expand**. For a table, the columns of the table are displayed. Pointing to the name of the table or of a column displays the name, application, type, path, and schema of the table or column. To collapse the display, click the Actions icon, then click **Collapse**.

You can increase or decrease the size of the displayed objects by using the + (plus) and - (minus) keys. You can reposition the objects by grabbing a blank spot in the display and dragging vertically or horizontally.

The lineage for some entities, such as analytic views, is more complex. An analytic view entity deployed for a business model has links to columns in a fact table and to hierarchies. The fact table has links to attribute dimensions and, for a data load job, to an ingest directive for the job. The ingest directive has a link to the source file. The attribute dimensions have links to tables for the dimensions. Those tables have links to ingest directives that have link to source files.

Impact

Impact shows all known information about the downstream use of an entity, and therefore how a change in the definition of an entity may affect other entities that depend on it. For example, if a table is used in an analytic view, a change to one of the column definitions in the table may affect the mapping from that column to the analytic view.

Classifications

For an analytic view and its attribute dimensions, hierarchies, and measures, the Classifications tab displays classifications and their values. Classifications are metadata that applications can use to present information about analytic views. When creating a business model, you may specify the values for the Caption and Description classifications.

Optimize

For an analytic view, the Optimize tab has information about caches created for the analytic view. A cache may exist if the advanced option Enable Autonomous Aggregate Cache was selected for the business model for which the analytic view is deployed.

Statistics

Statistics display information about the entity. For example, the statistics for a table include the size of the table and the numbers of rows and columns. They also include the names of the columns, their data types, the number of distinct values and null values, the maximum and minimum values, and other information.

The data is represented in the form of histogram which is column statistic which provides more detailed information about data distribution in a table's columns.

The histograms in the statistics pane can be representative of the following types:

- Frequency: In a frequency histogram, each distinct column value corresponds to a single bucket of the histogram. Since each value has its own dedicated bucket, some buckets may have many values, whereas others have few.
- Top-frequency: A top frequency histogram is a variation on a frequency histogram that ignores non-popular values that are statistically insignificant
- Height-Balanced: In this histogram, column values are divided into buckets so that each bucket contains approximately the same number of rows.
- Hybrid: A hybrid histogram combines characteristics of both height-based histograms and frequency histograms. This approach enables the optimizer to obtain better selectivity estimates in some situations.

Refer to Gathering Statistics for Tables section for more details.

Job Report

Displays a report of the total rows loaded and failed for a specific table.

The report displays of the total rows loaded and failed.

You can view the name of the table, the time the table was loaded and the time taken to process the load.

Data Definition

Data Definition displays the Oracle Autonomous Database DDL that created the entity.

You can view the following **Entity Types** from the drop-down:

- All
- Table
- View
- Analytic View
- Table,View,Analytic View
- Analytic Dashboard
- Attribute (Attribute Dimension)
- Attribute (Hierarchy)
- Attribute Dimension
- Cloud Link (Cloud Link Namespace)
- Cloud Link Namespace
- Cloud Object (Cloud Storage Link)
- Cloud Object (Data Catalog Asset)
- Cloud Object (Share Link)
- Cloud Storage Link
- Cloud Virtual Object (Cloud Storage Link)
- Column (Table)
- Column (View)
- Data Catalog Asset
- Data Catalog Link
- Database Link
- Directory
- External Location
- File (Directory)
- Function
- Hierarchy
- Ingest Job
- Level (Attribute Dimension)

- Level (Hierarchy)
- Live Table Feed
- Measure (Analytic View)
- Mining Model
- Package
- Procedure
- Schema
- Share
- Share Link
- Share Provider
- Share Recipient
- Share Schema (Share)
- Share Table (Share Schema)

Click the **Actions** icon next to **Table** entity to perform the following operations:

- Select **View Details** to view details about the table.
- Select Gather Statistics to display new statistics after you modify the table's structure. Refer to the Gathering Statistics for Tables for more detailed information.
- Select **Register to Cloud Link** to register the table for remote access for a selected audience you define. See Registering Cloud Links to Access Data section for more information.
- Select Create Analytic View to create an Analytic View from the selected associated table. See Creating Analytic Views
- Select Export Data to Cloud to export data to a cloud object store.
- Select **Edit** to edit the properties of the table. Refer to Editing Tables for more information.
- Select **Drop** to delete the table.

Click the **Actions** icon next to the **Share** entity to view the share entity details.

Click the **Actions** icon next to the **Share Provider** entity to view the share provider entity details.

Click the **Actions** icon next to the **Share Recipient** entity to view the share recipient entity details.

Click the **Actions** icon next to **Cloud Storage Link** entity to perform the following operations:

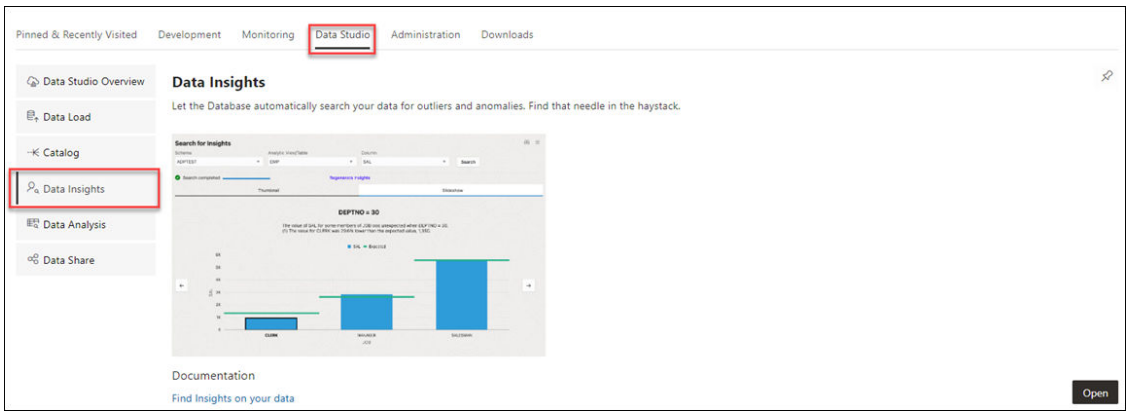
- Select **View Details** to view details about the table.
- Select **Objects** to view objects available in the selected storage link. You can click a file on the navigator pane to view it on the display area.
- Selecting **Link Tables** opens the Link Data page on the Data Load tool with the selected cloud storage link on the Cloud Location URL field. You can link data present in the cloud storage to the Autonomous Database. See Linking to Objects in Cloud Storage.
- Selecting **Load Tables** opens the Load Data page on the Data Load tool with the selected cloud storage link on the Cloud Location URL field. You can load data present in the cloud storage to the Autonomous Database. See Loading Data from Cloud Storage.

- Selecting **Create Live Table Feed** opens the Create Live Table feed wizard with he selected cloud storage link on the Cloud Location URL field. See Feeding Data to view more details.
- Select **Edit** to update any details on the cloud storage location. See Managing Connections to view details on creation of cloud storage location.
- Select **Rename** to rename the cloud store location to a different name.
- Select **Delete** to delete the cloud store location.

The Data Insights Page

The Data Insights page displays information about patterns and anomalies in the data of entities in your Oracle Autonomous Database.

To reach the Data Insights page, click the **Data Studio** tab in the Database Actions page, and select the **Data Insights** menu



or click the Selector icon and select **Data Insights** from the Data Tools menu in the navigation pane.

The following topics describe insights and how to generate and use them.

- About Insights
  You can generate insights for a table or for the analytic view deployed for data analysis.
- Generate Insights and View Reports
  Use these procedures to generate Insights and view reports about them.

About Insights

You can generate insights for a table or for the analytic view deployed for data analysis.

The insights that Data Insights generates for the analytic view of a business model can be more useful than those for a table because of the additional metadata that an analytic view provides.

Insights highlight data points as potentially anomalous if the actual value for a measure when filtering on pairs of analytic view hierarchy values or table column values is considerably higher or lower than the expected value, calculated across all hierarchy or column values. Insights highlight unexpected patterns, which you may want to investigate.

Insights are automatically generated by various analytic functions built into the database. The results of the insight analysis appear as a series of bar charts in the Data Insights dashboard.

Data Insights uses the following steps to generate insights:

1. Finds the values of a measure, for example Sales, across all of the distinct pairs of the hierarchy or column values for the measure. If Sales has the hierarchies or columns Marital Status, Age Band, Income Level, and Gender, then the pairs would be the values of each distinct value of each hierarchy or column paired to each distinct value of each of the other hierarchies or columns. For example, if the values of Marital Status are Married and Single, and the values of Age Band are A, B, and C, then the pairs would be Married and A, Married and B, Married and C, Single and A, Single and B, and Single and C. Each distinct value of Marital Status would also be paired with each distinct value of Income Level and Gender, and so on.
2. Estimates an expected value for the measure for each hierarchy or column pair.
3. Calculates the actual value for the measure for each hierarchy or column pair, for example Marital Status = S, Age Band = C, and then the difference between the actual value and the expected value.
4. Scores all of the differences and selects the largest variations between the actual and expected values to highlight as potential insights.

The resulting insights highlight cases where the measure value is significantly larger or smaller for a given hierarchy or column value pair than expected, for example much higher Sales where Marital Status = S and Age Band = C.

Insights for analytic views tend to use the higher levels of a hierarchy because the differences between the estimated and actual values are generally larger than they are for lower level attributes. For example, the difference in dollars between the estimated and actual sales for the entire USA are generally larger than the difference between the estimated and actual sales for a town with a population under 1000. The difference is calculated in absolute values, not percentages.

Insights for tables categorize columns as dimension columns or measure columns based on their data types and cardinality. A VARCHAR2 column is always categorized as a dimension, but a NUMBER column may be either a dimension or a measure. For example, a NUMBER column for YEAR values that has only 10 distinct values in a table with 1 million rows is assumed to be a dimension.

Generate Insights and View Reports

Use these procedures to generate Insights and view reports about them.

Generate Insights

To generate insights for a table or business model, do the following:

1. In the **Schema** field, select a schema.
2. In the **Analytic View/Table** field, select an analytic view or a table.
3. In the **Column** field, select a column that contains data about which you want gain insights.
4. Click **Search**.

A confirmation notice announces that the request for insights has been successfully submitted. Dismiss the notice by clicking the Close (X) icon in the notice.

A progress bar indicates that the search is in progress and when it has completed. The insights appear in the Data Insights dashboard as a series of bar charts.

To refresh the display of the insights, click **Refresh**. To have refreshes occur automatically, click **Enable Auto Refresh**.

Click **Recent Searches** to view the list of previous insights search.

Select **View Errors** to see any log of error that occurs while its creation. The results appear in a new browser tab.

View the Report

The charts in the Data Insights dashboard show the data that contain anomalous results. The bars in a chart show the actual values. The expected values are indicated by green horizontal lines. The bars that are outlined in black contain the most significant differences between the expected and the actual values.

For example, if the fact table for the insights records values about an insurance program, and the measures of the fact table are AGE_CODE, GENDER_CODE, INCOME_CODE, NUM_INSURED, NUM_UNINSURED, and YEAR, then insights might be generated for the NUM_INSURED measure. In that case, the dashboard would have a series of charts labeled YEAR and INCOME_CODE. Each chart would have a value of the related dimension in the upper left corner. For example, an INCOME_CODE chart that has a related AGE_CODE might have the AGE_CODE value 2 in the upper left corner.

Clicking a chart displays more details about it. At the top of the expanded view of the chart is the dimension name and value and a short textual analysis of notable insights. Below the analysis is the chart showing the values and insights about them.

For example, a chart for INCOME_CODE might have at AGE_CODE = 2 at the top, plus the textual analysis. In the chart, the INCOME_CODE values would be on the x-axis and the NUM_INSURED values would be on the y-axis. Pointing to a bar on the expanded chart displays the actual and the expected NUM_INSURED value for that INCOME_CODE and AGE_CODE.

Click the **Back** button to return to the Data Insights dashboard.

View Previous Reports

To see the results of a previous search, click the Recent Searches icon at the upper right. In the Recent Searches panel, click anywhere in the box for the insights search that you want to see.

To filter the previous searches, enter a value in the search field at the top of the Recent Searches panel.

To close the Recent Searches panel without selecting a search, click the X at the upper right of the panel.

Use Oracle OLAP with Autonomous Database

Oracle OLAP is available on Autonomous Database. Oracle OLAP supports uses such as planning, forecasting, budgeting, and analysis.

For more information, see the following:

- Getting Started with Oracle OLAP
- Oracle OLAP

Manage the Service

Describes basic tasks for managing an Autonomous Database instance.

- Lifecycle Operations
  You can start, stop, restart, rename, and terminate an Autonomous Database instance.
- Update Compute and storage limits
  Autonomous Database has the ability to increase its storage or CPU cores. You can also activate auto scaling.
- Update Billing Model, License Type, or Update Instance to a Paid Instance
  Describes how to view and update your license type and Oracle Database Edition for Autonomous Database. Also shows how to update the billing model to the ECPU compute model and shows how to update your instance from free (Always Free) to a paid instance.
- Cloning and Moving an Autonomous Database
- Use Refreshable Clones with Autonomous Database
  Autonomous Database provides cloning where you can choose to create a full clone of the active instance, create a metadata clone, or create a refreshable clone. With a refreshable clone the system creates a clone that can be easily updated with changes from the source database.
- Manage Autonomous Database Built-in Tools
  Autonomous Database includes built-in tools that you can enable and disable when you provision or clone a database, or at any time for an existing database.
- Use Autonomous Database Events
- Manage Time Zone File Updates on Autonomous Database
- Monitor Autonomous Database Availability
  Shows you the steps to view availability information for an Autonomous Database instance.
- Monitor Regional Availability of Autonomous Databases
  The regional availability is represented as a list of daily uptime percentages of Autonomous Databases across a set of services in each available region.

Lifecycle Operations

You can start, stop, restart, rename, and terminate an Autonomous Database instance.

- Provision Autonomous Database
  Follow these steps to provision a new Autonomous Database instance using the Oracle Cloud Infrastructure Console.
- Start Autonomous Database
  Describes the steps to start an Autonomous Database instance.
- Stop Autonomous Database
  Describes the steps to stop an Autonomous Database instance.
- Restart Autonomous Database
  Describes the steps to restart an Autonomous Database instance.
- Rename Autonomous Database
  Shows you the steps to change the database name for an Autonomous Database instance.

- Update the Display Name for an Autonomous Database Instance
  Shows you the steps to change the display name for an Autonomous Database instance.
- Terminate an Autonomous Database Instance
  Describes the steps to terminate an Autonomous Database instance.
- Change Autonomous Database Operation Mode to Read/Write Read-Only or Restricted
  You can select an Autonomous Database operation mode: Read/Write, Read-Only, or Restricted. The default mode is Read/Write.
- Schedule Start and Stop Times for an Autonomous Database Instance
  Describes the steps to schedule daily start and stop times for an Autonomous Database instance.
- Concurrent Operations on Autonomous Database
  Describes the actions that cause Oracle Cloud Infrastructure Console to ask you to confirm pausing or canceling a concurrent operation.

Provision Autonomous Database

Follow these steps to provision a new Autonomous Database instance using the Oracle Cloud Infrastructure Console.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- Choose your region. See Switching Regions for information on switching regions and working in multiple regions.
- Choose your **Compartment**. See Compartments for information on using and managing compartments.

On the Autonomous Databases page, perform the following steps:

1. Click **Create Autonomous Database**.
2. Provide basic information for the Autonomous Database.
   - **Compartment**. See Compartments for information on using and managing compartments.
   - **Display name** Specify a user-friendly description or other information that helps you easily identify the resource. The display name does not have to be unique.

     The default display name is a generated 16-character string that matches the default database name.
   - **Database name** Specify the database name; it must consist of letters and numbers only. The maximum length is 30 characters. The same database name cannot be used for multiple Autonomous Databases in the same tenancy in the same region.

     The default database name is a generated 16-character string that matches the default display name.
3. Choose a workload type. Select the workload type for your database from the choices:
   - **Data Warehouse**
   - **Transaction Processing**

- **JSON**
- **APEX**

4. Choose a deployment type.

   - **Serverless**
     Run Autonomous Database on serverless architecture.
   - **Dedicated Infrastructure**
     Run Autonomous Database on dedicated Exadata infrastructure.

   Select **Serverless** to create an Autonomous Database Serverless instance.

   See Create an Autonomous Database on Dedicated Exadata Infrastructure for steps to create your instance on dedicated Exadata infrastructure.

5. Configure the database (ECPU compute model)

   - **Always Free**: Select to show Always Free options.

     You can only create a free instance in the tenancy's Home region.

   - **Choose database version**: Select the database version. The available database version is 19c.

     With Always Free selected the available database versions are: Oracle Database 19c and Oracle Database 23ai.

   - **ECPU count**: Specify the number of CPUs for your database. The minimum value is 2.

     Your license type determines the **ECPU count** maximum. For example, if your license type is Bring your own license (BYOL) with Oracle Database Standard Edition (SE), the **ECPU count** maximum is 32.

   - **Compute auto scaling**: By default compute auto scaling is enabled to allow the system to automatically use up to three times more CPU and IO resources to meet workload demand. If you do not want to use compute auto scaling then deselect this option.

     See Use Auto Scaling for more information.

   - **Storage**: Specify the storage you wish to make available to your database. Depending on your workload type you have these options:

     – **Data Warehouse**: Specify your storage in Terabytes (TB).

     – **Transaction Processing**: Specify your storage in Gigabytes (GB) or Terabytes (TB). Enter the size in the **Storage** field. Select **GB** or **TB** for the **Storage unit size**.

   - By default, the IO capacity of your database depends on the number of ECPUs you provision. When you provision 384 TB of storage, your database is entitled to the full IO capacity of the Exadata infrastructure, independent of the number of ECPUs you provision.

     If you want to provision more than 384 TB of storage, file a Service Request at Oracle Cloud Support.

   - **Storage auto scaling**: By default storage auto scaling is disabled. Select if you want to enable storage auto scaling to allow the system to automatically expand to use up to three times more storage.

     See Use Auto Scaling for more information.

   - **Show advanced options**: Click to show the compute model options or if you want to create or join an elastic pool:

- **Enable elastic pool:**

  See Create or Join a Resource Pool While Provisioning or Cloning an Instance for more information.

- **Compute model**: Shows the selected compute model.

  Click **Change compute model** to change the compute model. After you select a different compute model, click **Save**.

  * **ECPU**
    Use the ECPU compute model for Autonomous Database. ECPUs are based on the number of cores elastically allocated from a pool of compute and storage servers.

  * **OCPU**
    Use the legacy OCPU compute model if your tenancy is using the OCPU model and you want to continue using OCPUs. The OCPU compute model is based on physical core of a processor with hyper threading enabled.

  > **Note:**
  >
  > OCPU is a legacy billing metric and has been retired for Autonomous Data Warehouse (**Data Warehouse** workload type) and Autonomous Transaction Processing (**Transaction Processing** workload type). Oracle recommends using ECPUs for all new and existing Autonomous Database deployments. See Oracle Support Document 2998742.1 for more information.

  See also Compute Models in Autonomous Database.

6. Backup retention

   Automatic backup retention period in days You have the option to select the automatic backup retention period, in a range from 1 to 60 days. You can restore and recover your database to any point-in-time in this retention period.

   This option is not available with the OCPU compute model.

   See About Backup and Recovery on Autonomous Database for more information.

7. Create administrator credentials. Set the password for the Autonomous Database Admin user.

   - **Username** This is a read only field.
   - **Password** Set the password for the Autonomous Database Admin user.
   - **Confirm password** Enter the same password again to confirm your new password.

   The password must meet the strong password complexity criteria based on Oracle Cloud security standards. For more information on the password complexity rules, see About User Passwords on Autonomous Database.

8. Choose network access

   > **Note:**
   >
   > After you provision your Autonomous Database you can change the network access option you select for the instance.

- **Secure access from everywhere**

  By default, secure connections are allowed from everywhere.

- **Secure access from allowed IPs and VCNs only**

  This option restricts connections to the database according to the access control lists (ACLs) you specify. To add multiple ACLs for the Autonomous Database, click **Add access control rule**.

  See Configure Access Control Lists When You Provision or Clone an Instance for more information.

- **Private endpoint access only**

  This option assigns a private endpoint, private IP, and hostname to your database. Specifying this option allows traffic only from the VCN you specify; access to the database from all public IPs or VCNs is blocked. This allows you to define security rules, ingress/egress, at the Network Security Group (NSG) level and to control traffic to your Autonomous Database.

  See Configure Private Endpoints When You Provision or Clone an Instance for more information.

9. Choose license and Oracle Database edition

   - **License included**

     The default is the license included license type. With this option you subscribe to new database software licenses and the database cloud service.

   - **Switch to Bring your own license (BYOL)**

     Select this to show more license options. Select this if your organization already owns Oracle Database software licenses and you want to bring your existing database software licenses to the database cloud service. See Cloud pricing for information on Bring your own license (BYOL) and other licensing options for Oracle Cloud Infrastructure cloud service pricing.

     When you select to switch to **Bring your own license (BYOL)**, you also select an Oracle Database edition. The Oracle Database edition you select is based on the license you bring to Autonomous Database and changes the maximum value that you can select for the **ECPU count**. The choices are:

     – **Oracle Database Enterprise Edition (EE)**: For this license type the maximum allowed value for **ECPU count** is 512, however you may contact your Oracle account team to request more ECPUs. With compute auto scaling enabled you can use up to **ECPU count** x 3 ECPUs. For example, if you set the **ECPU count** to 512, you can use up to 1,536 ECPUs.

     – **Oracle Database Standard Edition (SE)**: For this license type the maximum allowed value for **ECPU count** is 32. With compute auto scaling enabled you can use up to **ECPU count** x 3 ECPUs. This license restricts the number of ECPUs you can use to a maximum of 32 ECPUs, with or without compute auto scaling enabled.

       See Use Auto Scaling for more information.

10. (Optional) Provide contacts for operational notifications and announcements

    Click **Add contact** and in the **Contact email** field, enter a valid email address. To enter multiple **Contact email** addresses, repeat the process to add up to 10 customer contact emails.

    See View and Manage Customer Contacts for Operational Issues and Announcements for more information.

11. (Optional) Click Show advanced options to select advanced options.

    - **Encryption key**
      Encrypt using an Oracle-managed key: By default Autonomous Database uses Oracle-managed encryption keys. Using Oracle-managed keys, Autonomous Database creates and manages the encryption keys that protect your data and Oracle handles rotation of the TDE master key.

      Encrypt using a customer-managed key in this tenancy: If you this option, a master encryption key from a Oracle Cloud Infrastructure Vault in the same tenancy is used to generate the TDE master key on Autonomous Database.

      Encrypt using a customer-managed key located in a remote tenancy: If you this option, a master encryption key in the Oracle Cloud Infrastructure Vault located in a remote tenancy is used to generate the TDE master key on Autonomous Database.

      See Use Customer-Managed Encryption Keys on Autonomous Database for more information.

    - **Maintenance**
      Patch level By default the patch level is **Regular**. Select **Early** to configure the instance with the early patch level. Note: you cannot change the patch level after you provision an instance.

      See Set the Patch Level for more information.

    - **Management**
      Choose a character set and a national character set for your database.

      See Choose a Character Set for Autonomous Database for more information.

    - **Tools**
      If you want to view or customize the tools configuration, select the tools tab.

      See Configure Autonomous Database Built-in Tools when you Provision or Clone an Instance for more information.

    - **Tags**
      If you want to use Tags, enter the **Tag key** and **Tag value**. Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values which can be attached to resources.

      See Tagging Overview for more information.

12. Optionally, you can save the resource configuration as a stack by clicking **Save as Stack**. You can then use the stack to create the resource through the Resource Manager service.

    Enter the following details on the **Save as Stack** dialog, and click **Save**.

    - **Name:** Optionally, enter a name for the stack.
    - **Description:** Optionally, enter a description for this stack.
    - **Save in compartment:** Select a compartment where this Stack will reside.
    - **Tag namespace**, **Tag key**, and **Tag value**: Optionally, apply tags to the stack.

    For requirements and recommendations for Terraform configurations used with Resource Manager, see Terraform Configurations for Resource Manager. To provision the resources defined in your stack, apply the configuration.

13. Click **Create Autonomous Database**.

On the Oracle Cloud Infrastructure console the Lifecycle State shows **Provisioning** until the new database is available.

Start Autonomous Database

Describes the steps to start an Autonomous Database instance.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

1. On the **Details** page, from the **More actions** drop-down list, select **Start**.

   Start is only shown for a stopped instance.
2. Click **Start** to confirm.

Notes for starting a stopped Autonomous Database instance:

- When an Autonomous Database instance is started, Autonomous Database CPU billing is initiated, billed by the second with a minimum usage period of one minute.
- When an Autonomous Database remains stopped for over 7 days it may take a few minutes or longer to start up, depending on the size of the database. To avoid this extra startup time, Oracle recommends starting your Autonomous Database instance once every 7 days.

Stop Autonomous Database

Describes the steps to stop an Autonomous Database instance.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

1. On the **Details** page, from the **More actions** drop-down list, select **Stop**.
2. Click **Stop** to confirm.

   When concurrent operations such as scaling are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

Notes for a stopped Autonomous Database instance:

- Autonomous Database Tools are no longer able to connect to a stopped instance.
- Autonomous Database in-flight transactions and queries are stopped.
- Autonomous Database CPU billing is halted.

- When an Autonomous Database remains stopped for over 7 days it may take a few minutes or longer to start up, depending on the size of the database. To avoid this extra startup time, Oracle recommends starting your Autonomous Database instance once every 7 days.

Restart Autonomous Database

Describes the steps to restart an Autonomous Database instance.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

1. On the **Details** page, from the **More actions** drop-down list, select **Restart**.
2. In the confirmation dialog, select **Restart** to confirm.

   When concurrent operations such as scaling are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

The Autonomous Database instance state changes to "Restarting". After the restart is successful, Autonomous Database instance state is "Available".

> **Note:**
>
> When an Autonomous Database instance is restarted, Autonomous Database CPU billing is initiated, billed by the second with a minimum usage period of one minute.

Rename Autonomous Database

Shows you the steps to change the database name for an Autonomous Database instance.

Before you rename your database, note the following:

- A single tenancy in the same region cannot contain two Autonomous Databases with the same name.
- The database rename operation changes the connection strings required to connect to the database. Thus, after you rename a database you must download a new wallet for any existing instance wallets that you use to connect to the database (and any applications also must be updated to use the new connection string and the new wallet to connect to the database).

  If you are using a regional wallet, you can continue to use the existing wallet to connect to any databases that were not renamed. However, if you want to connect to a renamed database, you must download a new regional wallet.

  See Download Client Credentials (Wallets) for more information.

> **Note:**
>
> The rename operation terminates all connections to the database. After the rename operation completes, you can reconnect to the database.

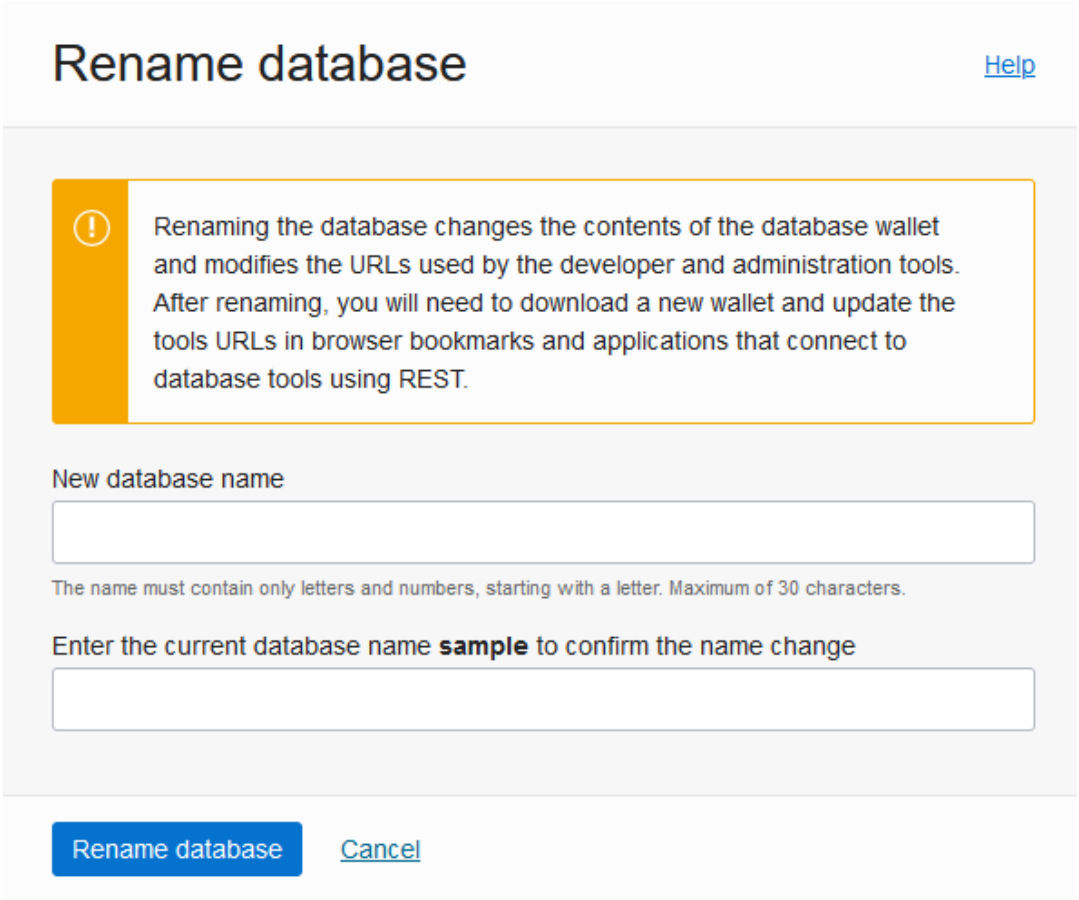
Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To rename an Autonomous Database instance, do the following:

1. On the **Details** page, from the **More actions** drop-down list, select **Rename Database**.

   This shows the Rename Database dialog.



2. On the Rename Database dialog, provide values for the fields:
   - New database name: the name you enter must consist of letters and numbers only. The maximum length is 30 characters.
   - Enter the current database name to confirm the name change

3. Click **Rename Database**.

After validating the fields, while the system renames the database the lifecycle state changes to **Updating**. You can start using the database after the rename operation completes and the lifecycle state shows **Available**.

Notes for Autonomous Database rename:

- Renaming your database does not change global references to your database from existing database links on remote databases. Changing such references is the responsibility of the administrator of the remote databases.
- When you rename a database, the Autonomous Database OCID does not change.
- If your database has Autonomous Data Guard enabled, the rename operation is not available.
- You cannot use the rename operation on a refreshable clone instance or on a database that is the source for a refreshable clone.
- The rename operation restarts the database.
- You can rename your Autonomous Database using the API. See UpdateAutonomousDatabase for more information.

Update the Display Name for an Autonomous Database Instance

Shows you the steps to change the display name for an Autonomous Database instance.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To update the display name on an Autonomous Database instance:

1. On the **Details** page, from the **More actions** drop-down list, select **Update display name**.

2. Enter a display name in the **New display name** field.
   - The display name minimum length is 1 character and maximum length is 255 characters.
   - The display name must include only letters, numbers, underscores "_", and hyphens "-".
   - The display name must start with a letter or an underscore "_", and cannot contain two successive hyphens "-".
3. Enter the current display name to confirm the change.
4. Click **Update display name**.

You can see the updated display name on the Oracle Cloud Infrastructure Console.

Terminate an Autonomous Database Instance

Describes the steps to terminate an Autonomous Database instance.

> **Note:**
>
> Terminating Autonomous Database permanently deletes the instance and removes all automatic backups. You cannot recover a terminated database.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

1. On the **Details** page, from the **More actions** drop-down list, select **Terminate**.
2. On the **Terminate Autonomous Database** page enter the database name to confirm that you want to terminate the database.

   When concurrent operations such as scaling are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.
3. Click **Terminate Autonomous Database**.

> **Note:**
>
> There are limitations for terminating a database when Autonomous Data Guard is enabled with a cross-region standby database. See Terminate a Cross-Region Standby Database for more information.

Change Autonomous Database Operation Mode to Read/Write Read-Only or Restricted

You can select an Autonomous Database operation mode: Read/Write, Read-Only, or Restricted. The default mode is Read/Write.

- Change Autonomous Database Operation Mode for a Session
  You can set the Autonomous Database operation mode for a session to Read-Only. When you set the session mode to Read-Only users in the session can only run queries.
- Change Autonomous Database Operation Mode to Read/Write Read-Only or Restricted
  From the Oracle Cloud Infrastructure Console you can select an Autonomous Database operation mode: Read/Write, Read-Only, or Restricted. The default mode is Read/Write.
- Autonomous Database Operation in Read-Only Mode
  When the database is in Read-Only mode, each of the following is disabled:

Change Autonomous Database Operation Mode for a Session

You can set the Autonomous Database operation mode for a session to Read-Only. When you set the session mode to Read-Only users in the session can only run queries.

To enable Read-Only mode for a session:

1. Connect to Autonomous Database.

   You can connect as the ADMIN user or as the user whose session you want to set to Read-only mode.
2. Run the following SQL statement:

```
ALTER SESSION SET READ_ONLY = TRUE;
```

To disable Read-Only mode for a session:

```
ALTER SESSION SET READ_ONLY = FALSE;
```

> **Note:**
>
> The `READ_ONLY` session parameter applies when the database is in Read/Write mode. If the database is in Read-Only mode, setting the session parameter value to `TRUE` or to `FALSE` does not change the operation mode for the session.

Change Autonomous Database Operation Mode to Read/Write Read-Only or Restricted

From the Oracle Cloud Infrastructure Console you can select an Autonomous Database operation mode: Read/Write, Read-Only, or Restricted. The default mode is Read/Write.

If you select Read-Only mode users can only run queries. In addition, for Read-Only or Read/Write mode, you can restrict access to only allow users with the `RESTRICTED SESSION` privilege to connect to the database. The ADMIN user has this privilege. You can use the restricted access mode to perform administrative tasks such as indexing, data loads, or other planned activities.

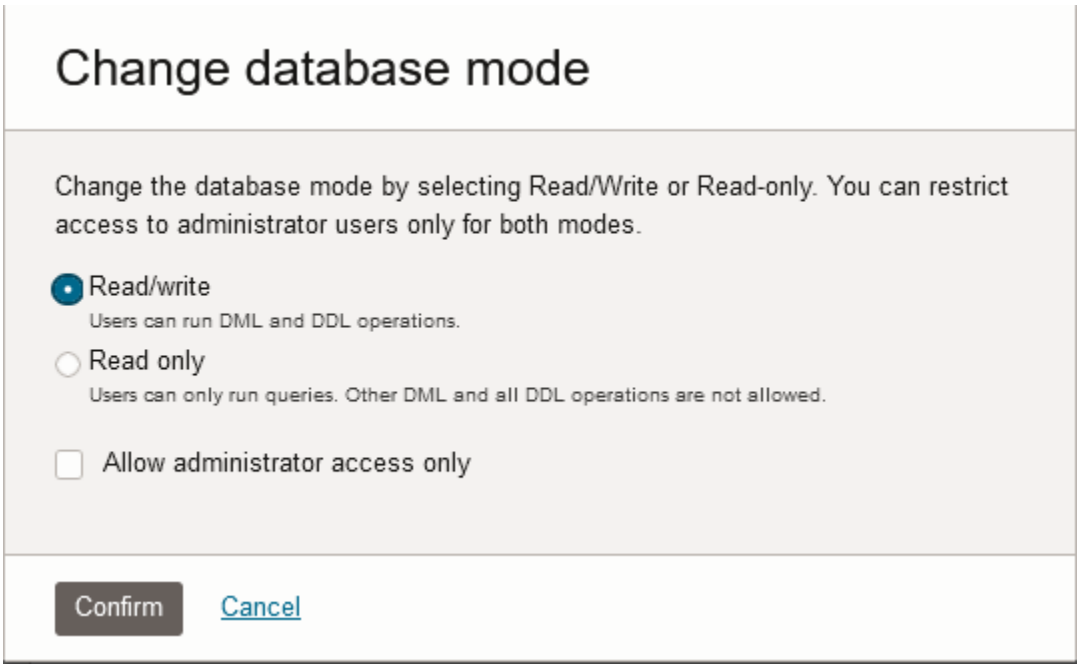
Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To change the Autonomous Database operation mode, do the following:

1. On the **Details** page, in the **Mode** field under General Information, click **Edit**.

   This shows the Change database mode dialog.



2. Select a mode, either **Read/write** or **Read only**.
3. If you want to restrict access to only allow the ADMIN or privileged users, then select **Allow administrator access only**.
4. Click **Confirm**.

   When concurrent operations such as scaling are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

   While the mode changes, the Lifecycle State field shows **Updating**. After the mode change completes, the Lifecycle State field shows **Available**.

Notes for changing the database mode:

- The database must be available with the Lifecycle State field showing **Available** to change the database mode.
- When you change the database mode to restricted mode or when you change from restricted mode to unrestricted mode, the database terminates existing database connections as follows:

  - When the mode is restricted, that is the Mode field shows **(Admin-only access)**, if you deselect **Allow administrator access only** and click **Confirm**, then users and applications need to reestablish database connections after the mode change completes.
  - When the mode is unrestricted, that is the Mode field does not show **(Admin-only access)**, if you select **Allow administrator access only** and click **Confirm**, then users and applications need to reestablish database connections after the mode change completes.

Autonomous Database Operation in Read-Only Mode

When the database is in Read-Only mode, each of the following is disabled:

- On the Oracle Cloud Infrastructure Console, the **Administrator Password** action is not allowed. Also, in Database Actions, under Administration, the **Database Users** card is disabled.
- Oracle APEX URLs are not available:
  - In Database Actions, under Development, the APEX link is disabled.
  - On the Oracle Cloud Infrastructure Console Tools tab, the **Open APEX** link is disabled.
- Oracle Machine Learning User Administration is disabled:
  - In Database Actions, under Development, the Oracle Machine Learning link is disabled.
  - On the Oracle Cloud Infrastructure Console Tools tab, the **Open Oracle ML User Administration** link is disabled.
- Setting resource management rules is disabled. In Database Actions, under Administration, the **Set Resource Management Rules** link is disabled.
- Upgrading the database is disabled when the database is in Read-Only mode.

Schedule Start and Stop Times for an Autonomous Database Instance

Describes the steps to schedule daily start and stop times for an Autonomous Database instance.

When you enable Auto Start/Stop Schedule on an Autonomous Database instance, the instance automatically starts and stops according to the schedule you specify. This allows you to reduce costs by scheduling shutdown periods for times when a system is not in use.

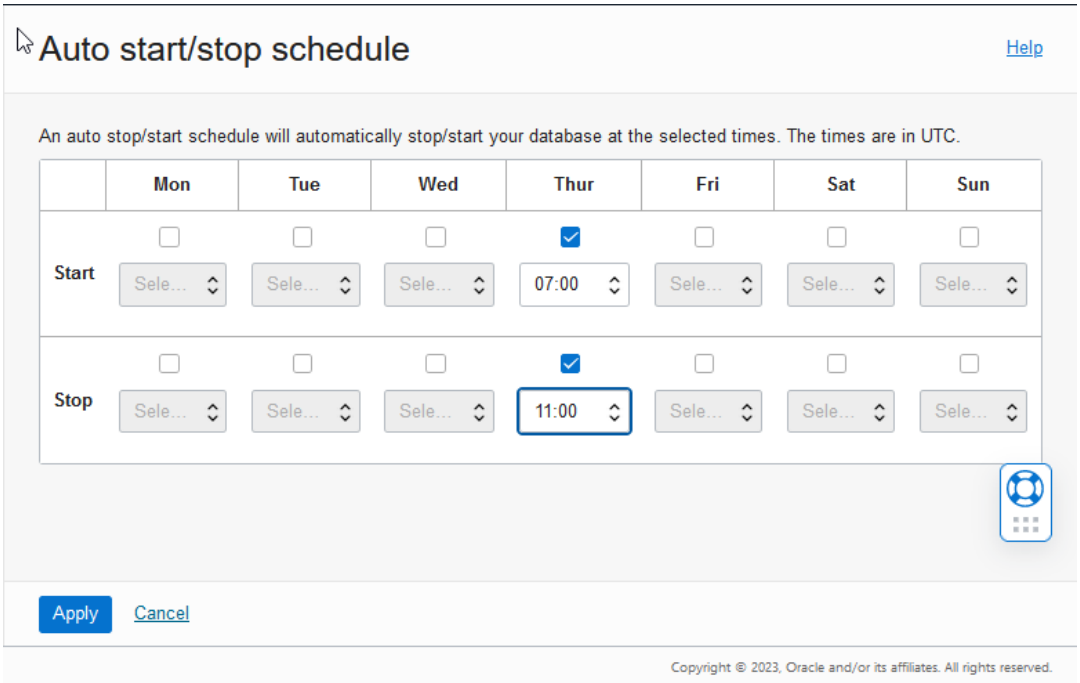
Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To set the Auto Start/Stop Schedule, do the following:

1. On the **Details** page, from the **More actions** drop-down list, select **Auto Start/Stop Schedule**.

2. For each day you want to schedule, select the start time check box and stop time check box and select a start time and a stop time.



   The drop-down list allows you to schedule start times and stop times hourly, on the hour, specified in Coordinated Universal Time (UTC).

   Selecting both a start time and a stop time is not required. You can select just a start time or just a stop time.

3. Click **Apply** to apply the schedule.
4. In the confirmation dialog, click **Apply**.

   While the system applies the Auto Start/Stop Schedule, the lifecycle state changes to **Updating**. When the operation completes the lifecycle state shows **Available**.

   The **Auto Start/Stop Schedule** field under General Information on the Autonomous Database Details page lists the days with an Auto Start/Stop Schedule enabled, and includes an **Edit** link.

To remove start and stop times for a day from an Auto Start/Stop Schedule, edit the schedule and deselect the schedule for that day. To remove all scheduled start and stop times and disable **Auto Start/Stop Schedule**, deselect the start and stop check boxes for all days in the schedule.

Notes for Autonomous Database Auto Start/Stop Schedule:

- A scheduled stop works the same way as a manual stop; when the scheduled stop occurs, any running database sessions are stopped.
- You can perform a manual start for a database, if necessary, after a database has stopped due to a scheduled stop.

  See Start Autonomous Database for more information.
- You can perform a manual stop for a database, if necessary, after a database has started due to a scheduled start.

  See Stop Autonomous Database for more information.
- The Lifecycle State must be **Available** to view or to edit the Auto Start/Stop Schedule.

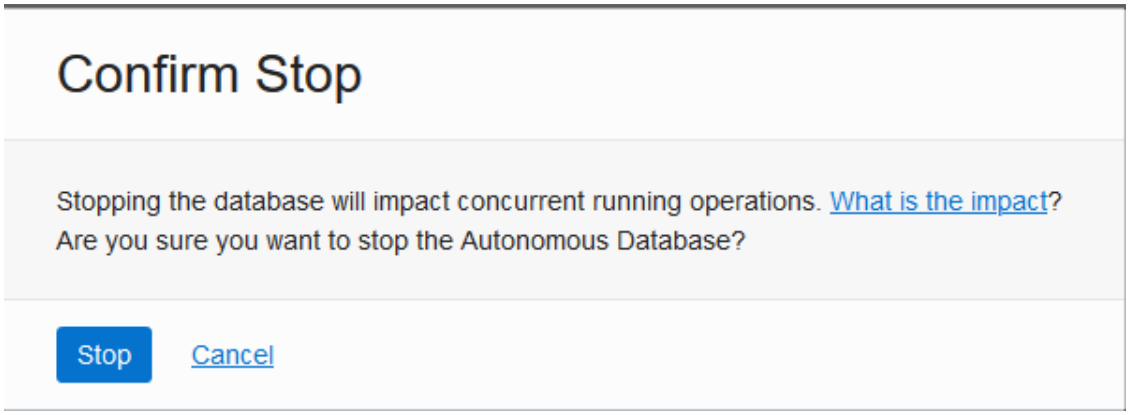
- Always Free Autonomous Database does not support Auto Start/Stop Scheduling.
- A database that has been scheduled stopped, remains stopped until the next scheduled start time or until you start the database. Likewise, a database that has been scheduled started, remains running until the next scheduled stop time or until you stop the database.
- When an Autonomous Database remains stopped for over 7 days it may take a few minutes or longer to start up, depending on the size of the database. To avoid this extra startup time, Oracle recommends starting your Autonomous Database instance once every 7 days.

Concurrent Operations on Autonomous Database

Describes the actions that cause Oracle Cloud Infrastructure Console to ask you to confirm pausing or canceling a concurrent operation.

When you initiate certain operations that take some time to complete, including scaling the system, these operations do not prevent you from performing other operations. For example, database connections, database operations, and most Autonomous Database actions proceed normally while you scale the system. However, certain database lifecycle management actions such as stopping the database have an impact on concurrent long-running operations.

For some actions a confirmation dialog displays to let you know when a concurrent operation is active. This allows you to proceed or to cancel the operation. For example, while stopping Autonomous Database you see a message such as the following:



- Manage Scaling with Concurrent Operations
  Describes the impact on scaling operations when you initiate certain actions during an ongoing scaling request.
- Long-Term Backup with Concurrent Operations
  Describes the impact on a long-term backup operation when you initiate certain actions during an ongoing long-term backup.

Manage Scaling with Concurrent Operations

Describes the impact on scaling operations when you initiate certain actions during an ongoing scaling request.

Action	Description
Stop	Stop pauses scaling. The scaling restarts when the database starts.
Restart	Restart pauses scaling. The scaling restarts when the database starts.

Action	Description
Restore	Restore pauses scaling. The scaling restarts when the restore completes if resources allow.
Switchover	Switchover pauses scaling. The scaling restarts when the switchover completes. The scaling also occurs on the standby database after the switchover.
Failover	Failover pauses scaling. The scaling restarts when the failover completes. The scaling also occurs on the standby database if it is available after the failover.
Terminate	Scaling stops and the database is terminated.
Change Database Mode Read-Write or Read-Only	A mode change pauses scaling. The scaling restarts when the database starts after the mode change completes.

Long-Term Backup with Concurrent Operations

Describes the impact on a long-term backup operation when you initiate certain actions during an ongoing long-term backup.

Action	Description
Stop	Stop cancels an ongoing long-term backup. The long-term backup does not restart when the database starts.
Restart	Restart cancels an ongoing long-term backup. The long-term backup does not restart when the database starts.
Restore	Restore cancels an ongoing long-term backup.
Switchover	Switchover cancels an ongoing long-term backup. The long-term backup does not restart after the switchover completes.
Failover	Failover cancels an ongoing long-term backup. The long-term backup does not restart after the failover completes.
Terminate	Terminate cancels an ongoing long-term backup.

Update Compute and storage limits

Autonomous Database has the ability to increase its storage or CPU cores. You can also activate auto scaling.

- Add CPU or Storage Resources or Enable Auto Scaling
  Describes how to scale your Autonomous Database on demand by adding CPU cores or storage. Also describes how to enable auto scaling.
- Remove CPU or Storage Resources or Disable Auto Scaling
  Describes how to scale your Autonomous Database on demand by removing CPU cores or storage. Also describes how to disable auto scaling.
- Use Auto Scaling
  When you create an Autonomous Database instance, by default compute auto scaling is enabled and storage auto scaling is disabled. You can manage auto scaling from the Oracle Cloud Infrastructure Console to enable or disable compute auto scaling or storage auto scaling.

Add CPU or Storage Resources or Enable Auto Scaling

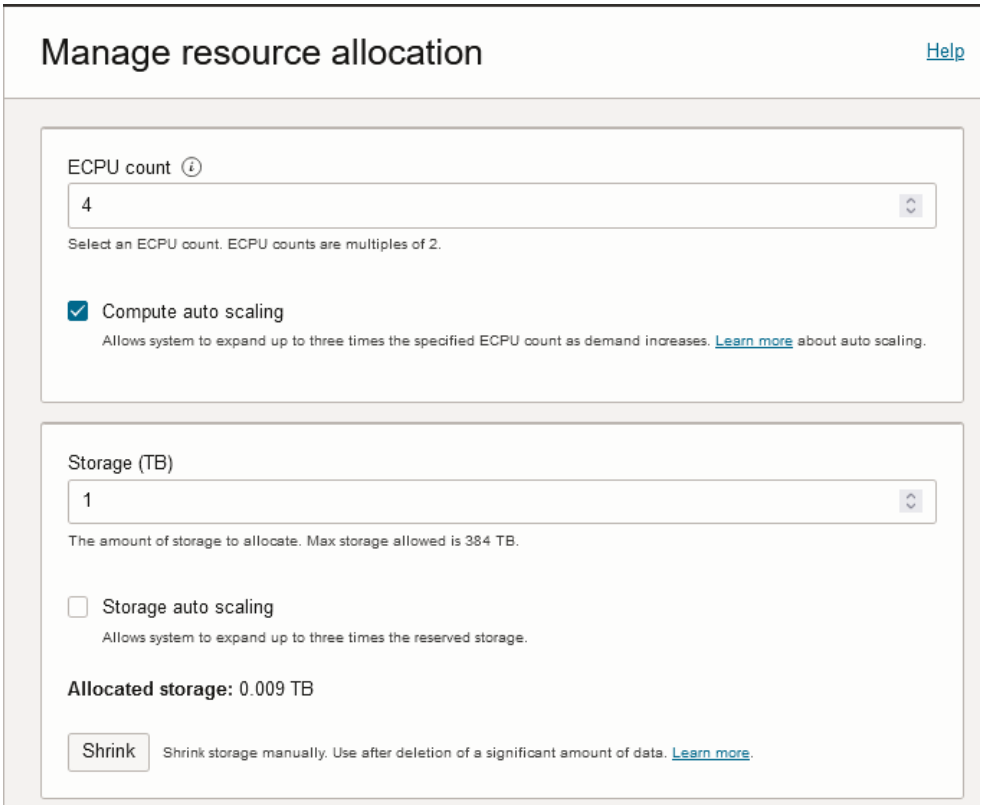
Describes how to scale your Autonomous Database on demand by adding CPU cores or storage. Also describes how to enable auto scaling.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

1. On the **Details** page, click **Manage resource allocation**.
2. In the **Manage resource allocation** area, add resources for your scale request.
   - Enter a value or click the up arrow to select a value for **ECPU count** (**OCPU count** if your database uses OCPUs). The default is no change.
   - **Storage**: Specify the storage you wish to make available to your database. Depending on your workload type and your compute model, you have these options:
     - **Data Warehouse**: Specify your storage in Terabytes (TB).
     - **JSON**: Specify your storage in Terabytes (TB).
     - **Transaction Processing**: Specify your storage in Gigabytes (GB) or Terabytes (TB). Enter the size in the **Storage** field. Select **GB** or **TB** for the **Storage unit size**. GB units are only available when the Workload type is Transaction Processing and the Compute model is **ECPU**.

     The default is no change.



3. Enable compute auto scaling.

   Select **Compute auto scaling** to allow the system to automatically use up to three times more CPU and IO resources to meet workload demand, compared to the database operating with compute auto scaling disabled. The additional resources are available until you disable compute auto scaling.

   If compute auto scaling is disabled while more CPUs are in use than the specified **ECPU count** (**OCPU count** if your database uses OCPUs), then Autonomous Database scales the number of CPUs in use down to the **ECPU count** (**OCPU count**).

   See Use Auto Scaling for more information.

4. To enable Storage auto scaling, select **Storage auto scaling**.

   When you select **Storage auto scaling** the database can expand to use up to three times the reserved base storage.

   If you disable **Storage auto scaling** and the used storage is greater than the reserved base storage, as specified by the storage shown in the **Storage** field on the Oracle Cloud Infrastructure Console, Autonomous Database shows a warning on the disable storage auto scaling confirmation dialog. The warning lets you know that the reserved base storage value will be increased to the nearest TB greater than the actual storage usage, and shows the new reserved base storage value.

   See Use Auto Scaling for more information.

> **Note:**
>
> Clicking **Shrink** initiates the Shrink storage operation. See Shrink Storage for more information.

5. Click **Apply** to change your resources.

When you click **Apply** with a resource change, the Lifecycle State changes to **Scaling in Progress...**. After the Lifecycle State changes to **Available** the changes apply immediately.

Remove CPU or Storage Resources or Disable Auto Scaling

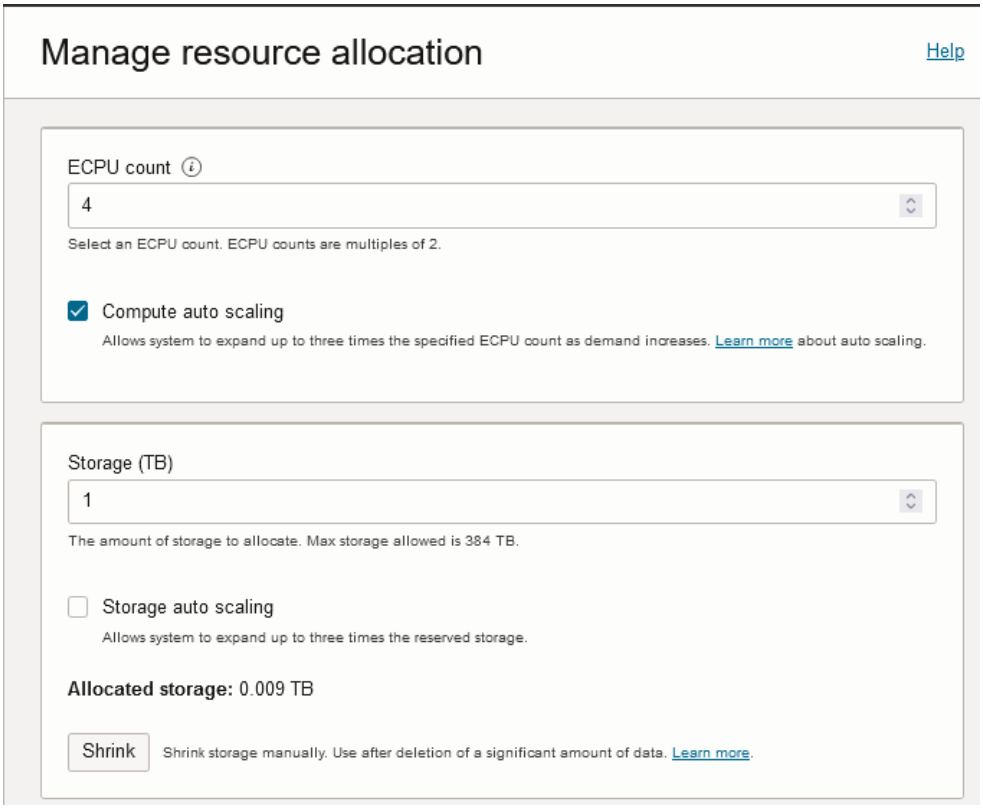
Describes how to scale your Autonomous Database on demand by removing CPU cores or storage. Also describes how to disable auto scaling.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

1. On the **Details** page, click **Manage resource allocation**.
2. In the **Manage resource allocation** area, select the change in resources for your scale request:
   - Enter a value or click the down arrow to select a value for the **ECPU count** (**OCPU count** if your database uses OCPUs). The default is no change.
   - **Storage**: Specify the storage you wish to make available to your database. Depending on your workload type and your compute model, you have these options:
     - **Data Warehouse**: Specify your storage in Terabytes (TB).
     - **JSON**: Specify your storage in Terabytes (TB).
     - **Transaction Processing**: Specify your storage in Gigabytes (GB) or Terabytes (TB). Enter the size in the **Storage** field. Select **GB** or **TB** for the **Storage unit size**. GB units are only available when the Workload type is Transaction Processing and the Compute model is **ECPU**.

     The default is no change.



3. When compute auto scaling is enabled, deselect **Compute auto scaling** to disable compute auto scaling.

   If compute auto scaling is disabled while more CPUs are in use than the specified **ECPU count** (**OCPU count** if your database uses OCPUs), then Autonomous Database scales the number of CPUs in use down to the **ECPU count** (**OCPU count** if your database uses OCPUs).

   See Use Auto Scaling for more information.

4. When Storage auto scaling is enabled, deselect **Storage auto scaling** to disable storage auto scaling.

   If you disable **Storage auto scaling** and the used storage is greater than the reserved base storage, as specified by the storage shown in the **Storage** field on the Oracle Cloud Infrastructure Console, Autonomous Database shows a warning on the disable storage auto scaling confirmation dialog. The warning lets you know that the reserved base storage value will be increased to the nearest TB greater than the actual storage usage, and shows the new reserved base storage value.

   See Use Auto Scaling for more information.

> **Note:**
>
> Clicking **Shrink** initiates the Shrink storage operation. See Shrink Storage for more information.

5. Click **Apply** to change your resources.

When you click **Apply** with a resource change, the Lifecycle State changes to **Scaling in Progress...**. After the Lifecycle State changes to **Available** the changes apply immediately.

Note the following restrictions for scaling down storage:

- Scaling down storage is not allowed if the Autonomous Database instance contains either of the following:
  - Advanced Queuing tables
  - Tables with `LONG` columns
- If you have columns with the `ROWID` data type, the `ROWID`s that these column values point to may change during the scale down storage operation.
- Tables that contain the following may be moved offline during a scale down operation. DML operations on these tables may be blocked for the duration of the move and the table indexes for these tables may become unusable until the scale down operation completes:
  - Tables with bitmap join indexes
  - Nested tables
  - Object tables
  - Partitioned tables with domain indexes

Use Auto Scaling

When you create an Autonomous Database instance, by default compute auto scaling is enabled and storage auto scaling is disabled. You can manage auto scaling from the Oracle Cloud Infrastructure Console to enable or disable compute auto scaling or storage auto scaling.

- Compute Auto Scaling
- Storage Auto Scaling
  When you create an Autonomous Database instance, by default **Storage auto scaling** is disabled. You can manage scaling and enable storage auto scaling from the Oracle Cloud Infrastructure Console or using the API.
- Shrink Storage
  When the storage used in the database is significantly lower than the allocated storage, the shrink operation reduces the allocated storage.

Compute Auto Scaling

With compute auto scaling enabled the database can use up to three times more CPU and IO resources than specified by the number of ECPUs (OCPUs if your database uses OCPUs) as shown in the **ECPU count** or **OCPU count** field on the Oracle Cloud Infrastructure Console.

When auto scaling is enabled, if your workload requires additional CPU and IO resources, the database automatically uses the resources without any manual intervention required. For example:

- In the ECPU compute model, when the **ECPU count** is 512, this allows the database to use up to 512 x 3 ECPUs (1536 ECPUs) when auto scaling is enabled.

  To see the average number of ECPUs used during an hour you can use the "Number of ECPUs allocated" graph on the Overview tab on the **Database Dashboard** card in Database Actions. See Database Dashboard Overview for more information.
- In the OCPU compute model, when the **OCPU count** is 128, this allows the database to use up to 128 x 3 OCPUs (384 OCPUs) when auto scaling enabled.

To see the average number of OCPUs used during an hour you can use the "Number of OCPUs allocated" graph on the Overview tab on the **Database Dashboard** card in Database Actions. See Database Dashboard Overview for more information.

Enabling compute auto scaling does not change the concurrency and parallelism settings for the predefined services. See Manage Concurrency and Priorities on Autonomous Database for more information.

> **Note:**
>
> Your license type determines the **ECPU count** maximum. For example, if your license type is Bring your own license (BYOL) with Oracle Database Standard Edition (SE), the **ECPU count** maximum is 32.For this license type the maximum allowed value for **ECPU count** is 32. With compute auto scaling enabled you can use up to **ECPU count** x 3 ECPUs. This license restricts the number of ECPUs you can use to a maximum of 32 ECPUs, with or without compute auto scaling enabled.

When compute auto scaling is enabled your database may use and you may be billed for additional CPU consumption as needed by your workload, up to three times (3x) the number of base CPUs (as shown in the **ECPU count** or **OCPU count** field on the Oracle Cloud Infrastructure Console). See Oracle Autonomous Database Serverless Features Billing for details on compute auto scaling billing.

See Add CPU or Storage Resources or Enable Auto Scaling for the steps to enable compute auto scaling.

Storage Auto Scaling

When you create an Autonomous Database instance, by default **Storage auto scaling** is disabled. You can manage scaling and enable storage auto scaling from the Oracle Cloud Infrastructure Console or using the API.

With **Storage auto scaling** enabled, the Autonomous Database can expand to use up to three times the reserved base storage, as specified by the storage shown in the **Storage** field on the Oracle Cloud Infrastructure Console. If you need additional storage, the database automatically uses the reserved storage without any manual intervention required.

You specify the base storage when you provision or clone your database, or you can change the storage at any time by clicking **Manage resource allocation** and changing the storage size. Depending on your workload type and the compute model selection, you have these options to specify the reserved base storage units:

- **Data Warehouse**: Specify your storage in Terabytes (TB).
- **Transaction Processing**: Specify your storage in Gigabytes (GB) or Terabytes (TB). GB units are only available when the Workload type is Transaction Processing and the Compute model is ECPU.

Using the Oracle Cloud Infrastructure Console or the APIs you can provision or auto-scale an Autonomous Database instance's storage up to 384 TB. For storage requirements larger than 384 TB, Oracle recommends that you file a Service Request at Oracle Cloud Support.

For example, if your storage is 100 TB and storage auto scaling is enabled, you have access to a maximum of 300 TB of storage and if your storage is 200 TB, you have access to a maximum of 384 TB (if you requested a larger maximum by filing a service request, then the maximum would be your custom maximum size).

As data flows in, you are billed as follows:

- For storage usage below your reserved base storage, you are billed based on your base storage.
- After your allocated storage exceeds your reserved base storage, storage usage is billed based on your allocated storage rounded up to the nearest TB, in a given hour.

For example, if your reserved base storage is 4 TB, until your allocated storage exceeds 4TB of storage, you are billed based on your base storage (4 TB). After you exceed 4 TB, storage is billed based on the allocated storage rounded up to the nearest TB, in a given hour. In this example, if the allocated storage grows over 4 TB in a given hour, say to 4.9 TB, you are billed for 5 TB of storage from that hour onward.

If you then delete 1 TB of data, your allocated storage remains at 4.9 TB and you are billed for 5 TB until you perform a shrink operation. When you perform a shrink operation, Autonomous Database may be able to reduce your allocated storage back to 3.9TB (shrinking the data and undo tablespaces). After the shrink operation completes and your allocated storage (3.9TB) is once again below your reserved base storage (4 TB), you will once again be billed for your reserved base storage of 4 TB. See Shrink Storage for more information.

> **Note:**
>
> Reducing temp tablespace requires a database restart.

If you disable **Storage auto scaling** and the used storage is greater than the reserved base storage, as specified by the storage shown in the **Storage** field on the Oracle Cloud Infrastructure Console, Autonomous Database shows a warning on the disable storage auto scaling confirmation dialog. The warning lets you know that the reserved base storage value will be increased to the nearest TB greater than the actual storage usage, and shows the new reserved base storage value.

To see the Autonomous Database instance storage usage, you can view the "Storage allocated" and "Storage used" graphs on the Overview tab by clicking the **Database Dashboard** card in Database Actions. See Database Dashboard Overview for more information.

See Add CPU or Storage Resources or Enable Auto Scaling for the steps to enable storage auto scaling.

Shrink Storage

When the storage used in the database is significantly lower than the allocated storage, the shrink operation reduces the allocated storage.

To understand storage allocation and the shrink operation, note the following:

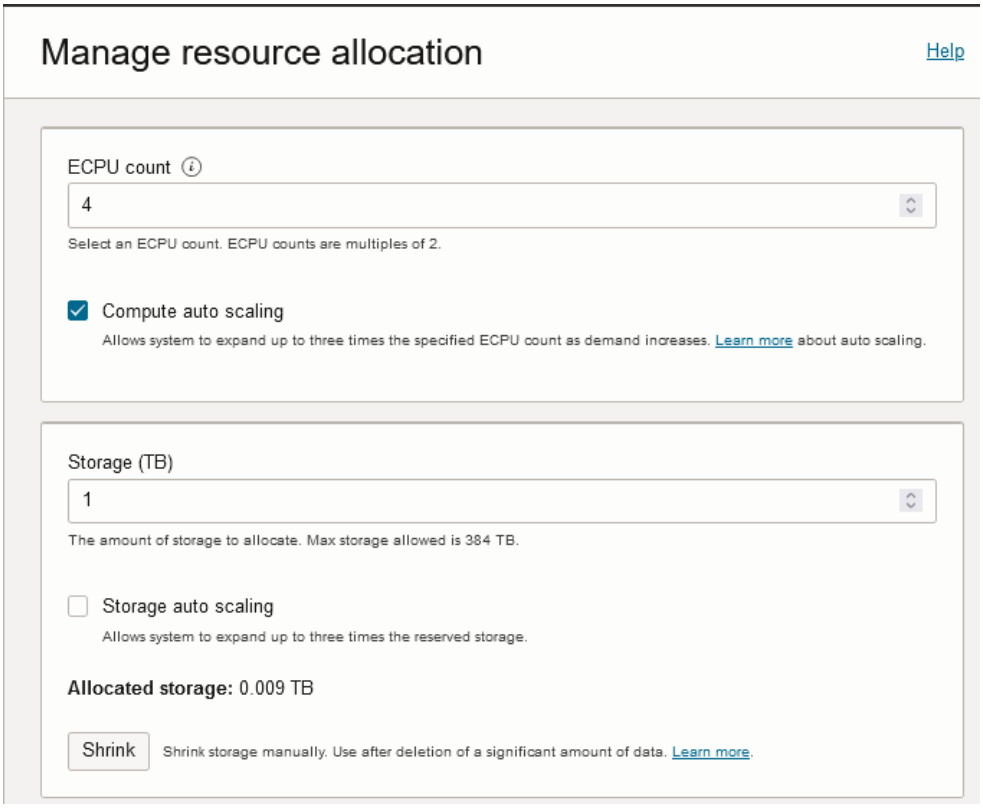
- Reserved base storage: is the base amount of storage you select for the database when you provision or scale the database, excluding any auto-scaled value. The reserved base storage is shown in the **Storage** field on the Oracle Cloud Infrastructure Console.
- Allocated storage: is the amount of storage physically reserved for all database tablespaces (excluding sample schema tablespaces). This number also includes the free space in these tablespaces.
- Used storage: is the amount of storage actually used in all tablespaces (excluding the sample schema tablespaces). The used storage excludes the free space in these tablespaces. Used storage is the storage actually used by database objects, tables, indexes, and so on, including internally used temp space.

- Maximum storage: is the maximum storage reserved. When storage auto scaling is disabled, the maximum storage equals the reserved base storage. When storage auto scaling is enabled, the maximum storage is three times the base storage (maximum = reserved base x 3).

> **Note:**
>
> The **Shrink** operation is not available with Always Free Autonomous Database.

To shrink storage:

1. On the **Details** page, click **Manage resource allocation**.
2. In the **Manage resource allocation** area, select **Shrink**.



3. Click **Confirm** in the Shrink Database dialog.

> **Note:**
>
> The **Shrink** operation is a long running operation.

The **Shrink** operation requires that all of the following apply:

- **Storage auto scaling** must be enabled.
- The allocated storage must be greater than the reserved base storage.

- The allocated storage, rounded up to the nearest 1TB, can be reduced by 1TB or more.
- The following must be true:

  Allocated storage - Used storage > 100 GB

When you click **Shrink** and these conditions are not met, Autonomous Database shows the **Action unavailable** dialog.

Note the following for the Shrink operation:

- The shrink operation runs an `alter table... move online` operation which uses the database's CPUs. In cases where the shrink operation is running slow or taking a very long time, Oracle recommends that you scale up the number of CPUs. See Add CPU or Storage Resources or Enable Auto Scaling for more information.
- The shrink operation is not allowed if the Autonomous Database instance contains either of the following:
  - Advanced Queuing tables
  - Tables with `LONG` columns
- If you have columns with the `ROWID` data type, the `ROWID`s that these column values point to may change during the shrink operation.
- Tables that contain the following may be moved offline during the shrink operation. DML operations on these tables may be blocked for the duration of the move and the table indexes for these tables may become unusable until the shrink operation completes:
  - Tables with bitmap join indexes
  - Nested tables
  - Object tables
  - Immutable tables
  - Blockchain tables
  - Partitioned tables with domain indexes
- If you perform a **Shrink** operation very soon after a data deletion operation, the **Shrink** operation may fail. This can be due to the delay required for Autonomous Database to recalculate storage values. In this case, Oracle recommends that you retry the **Shrink** operation (that is, wait for several minutes for the storage deletion and any associated storage usage updates to complete and perform the **Shrink** operation again).

Update Billing Model, License Type, or Update Instance to a Paid Instance

Describes how to view and update your license type and Oracle Database Edition for Autonomous Database. Also shows how to update the billing model to the ECPU compute model and shows how to update your instance from free (Always Free) to a paid instance.

- View and Update Your License and Oracle Database Edition on Autonomous Database (ECPU Compute Model)
- View and Update Your License and Oracle Database Edition on Autonomous Database (OCPU Compute Model)
- Update to ECPU Billing Model on Autonomous Database
- Update Always Free Instance to Paid with Autonomous Database
  Describes how to update your instance to paid from free with Autonomous Database.

- Upgrade Autonomous JSON Database to Autonomous Transaction Processing
  You can promote an Autonomous JSON Database to an Autonomous Transaction Processing database at any time.
- Upgrade APEX Service to Autonomous Transaction Processing
  Easily upgrade APEX Service to Autonomous Transaction Processing.

View and Update Your License and Oracle Database Edition on Autonomous Database (ECPU Compute Model)

Describes how to view and update your license type and Oracle Database Edition for Autonomous Database.

> **Note:**
>
> See View and Update Your License and Oracle Database Edition on Autonomous Database (OCPU Compute Model) if you are using the OCPU compute model.

The **License type** field on the Autonomous Database Information tab shows your license and Oracle Database Edition. For example:

License type: Bring Your Own License (BYOL), Enterprise Edition

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To change your license type or Oracle Database Edition:

1. On the **Details** page, from the **More actions** drop-down list, select **Update license and Oracle Database edition**.
2. On the **Update license and Oracle Database edition** page select a license and Oracle Database Edition:
   - **Bring your own license (BYOL)** Select if your organization already owns Oracle database software licenses. Bring your existing database software licenses to the database cloud service.
   - Choose an Oracle Database Edition:

     When you select **Bring your own license (BYOL)**, you also choose an Oracle Database Edition. The edition you select is based on the license you bring to Autonomous Database and changes the maximum value that you can select for the **ECPU count**.The choices are:

     Oracle Database Enterprise Edition (EE): For this license type the maximum allowed value for **ECPU count** is 512, however you may contact your Oracle account team to request more ECPUs. With compute auto scaling enabled you can use up to **ECPU count** x 3 ECPUs. For example, if you set the **ECPU count** to 512, you can use up to 1,536 ECPUs.

Oracle Database Standard Edition (SE): For this license type the maximum allowed value for **ECPU count** is 32. With compute auto scaling enabled you can use up to **ECPU count** x 3 ECPUs. This license restricts the number of ECPUs you can use to a maximum of 32 ECPUs, with or without compute auto scaling enabled.

See Use Auto Scaling for more information.

- **License included**

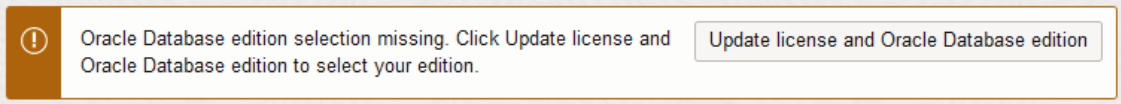
  Subscribe to new database software licenses and the database cloud service.

3. Click **Save Changes**.

While the system applies the changes, the lifecycle state changes to **Updating**. The database remains up and accessible, there is no downtime while the license type updates. When the operation completes the lifecycle state shows **Available**.

d does not require downtime.

If you have not specified a complete license type, Autonomous Database instances created before the introduction of the Oracle Database Edition will see the following message on the Oracle Cloud Infrastructure Console:



To dismiss this message, click **Update license and Oracle Database edition** and set a license type and Oracle Database Edition.

For information on Bring your own license (BYOL) and other licensing options for Oracle Cloud Infrastructure, see the following:

- Lower your TCO for Oracle Database Standard Edition with the Bring Your Own License (BYOL) program
- For BYOL policies, see Oracle Cloud Services contracts. BYOL policies are described in the **Oracle PaaS and IaaS Universal Credits Service Descriptions** document under **Cloud Service Descriptions**.
- Frequently asked questions: Oracle Bring Your Own License (BYOL)
- Cloud pricing

Notes for performing Update license and Oracle Database edition operation:

- If you try to switch to License type: Bring your own license (BYOL) Standard Edition when the base ECPU count is above 32, you see the following message:
  ```
  You cannot select Oracle Database Standard Edition unless your ECPU count is a total of sixteen (32) or less ECPUs. Use the Manage Scaling option to adjust your ECPU count before changing to Standard Edition.
  ```
  In this case, lower your ECPU count before you change the license type. See Remove CPU or Storage Resources or Disable Auto Scaling for more information.
- If you switch to License type: Bring your own license (BYOL) Standard Edition when the base ECPU count is 32 and compute auto scaling is enabled, you see the following message:
  ```
  Compute auto scaling will be disabled with a base ECPU count of 32, under an Oracle Database Standard Edition license.
  ```
  In this case, **Compute auto scaling** is disabled and the **ECPU count** is set to 32.

- If you switch to License type: Bring your own license (BYOL) Standard Edition when the base ECPU count is below 32 and above 8, with compute auto scaling enabled, you see the following message:
  `Your compute auto scaling maximum will be set to 32 ECPUs, under an Oracle Database Standard Edition license.`

  In this case, the maximum number of ECPUs with compute auto scaling is 32.

View and Update Your License and Oracle Database Edition on Autonomous Database (OCPU Compute Model)

Describes how to view and update your license type and Oracle Database Edition for Autonomous Database.

> **Note:**
>
> See View and Update Your License and Oracle Database Edition on Autonomous Database (ECPU Compute Model) if you are using the ECPU compute model.

The **License Type** field on the Autonomous Database Information tab shows your license and Oracle Database Edition. For example:

License Type: Bring Your Own License (BYOL), Enterprise Edition

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To change your license type or Oracle Database Edition:

1. On the **Details** page, from the **More actions** drop-down list, select **Update license and Oracle Database edition**.
2. On the **Update license and Oracle Database edition** page select a license and Oracle Database Edition:
   - **Bring Your Own License (BYOL)** Select if your organization already owns Oracle database software licenses. Bring your existing database software licenses to the database cloud service.
   - Choose an Oracle Database Edition:

     When you select **Bring your own license (BYOL)**, you also choose an Oracle Database Edition. The choices are:

     Oracle Database Enterprise Edition (EE): For this license type the maximum allowed value for **OCPU count** is 128, however you may contact your Oracle account team to request more than 128 OCPUs. With auto scaling enabled you can use up to **OCPU count** x 3 OCPUs. For example, if you set the **OCPU count** to 128, you can use up to 384 OCPUs.

Oracle Database Standard Edition (SE): For this license type the maximum allowed value for **OCPU count** is 8. With auto scaling enabled you can use up to **OCPU count** x 3 OCPUs. This license restricts the number of OCPUs you can use to a maximum of 8 OCPUs, with or without auto scaling enabled.

The edition you select is based on the license you bring to Autonomous Database and changes the maximum value that you can select for the **OCPU count**.

See Use Auto Scaling for more information.

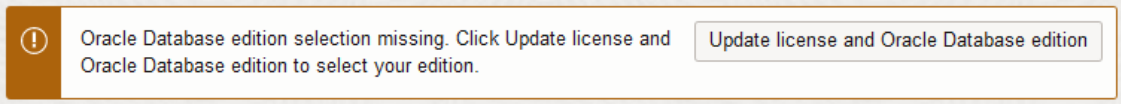
- **License included**

  Subscribe to new database software licenses and the database cloud service.

3. Click **Save Changes**.

While the system applies the changes, the lifecycle state changes to **Updating**. The database remains up and accessible, there is no downtime while the license type updates. When the operation completes the lifecycle state shows **Available**.

If you have not specified a complete license type, Autonomous Database instances created before the introduction of the Oracle Database Edition will see the following message on the Oracle Cloud Infrastructure Console:



To dismiss this message, click **Update license and Oracle Database edition** and set a license type and Oracle Database Edition.

For information on Bring Your Own License (BYOL) and other licensing options for Oracle Cloud Infrastructure, see the following:

- Lower your TCO for Oracle Database Standard Edition with the Bring Your Own License (BYOL) program
- For BYOL policies, see Oracle Cloud Services contracts. BYOL policies are described in the **Oracle PaaS and IaaS Universal Credits Service Descriptions** document under **Cloud Service Descriptions**.
- Frequently asked questions: Oracle Bring Your Own License (BYOL)
- Cloud pricing

Notes for performing Update license and Oracle Database edition operation:

- If you try to switch to License Type: Bring your own license (BYOL) Standard Edition when the base OCPU count is above 8, you see the following message:

  ```
  You cannot select Oracle Database Standard Edition unless your OCPU count is a total of eight (8) or less OCPUs. Use the Manage Scaling option to adjust your OCPU count before changing to Standard Edition.
  ```

  In this case, lower your OCPU count before you change the license type. See Remove CPU or Storage Resources or Disable Auto Scaling for more information.

- If you switch to License Type: Bring your own license (BYOL) Standard Edition when the base OCPU count is 8 and **Compute auto scaling** is enabled, you see the following message:

  ```
  Compute auto scaling will be disabled with a base OCPU count of 8, under an Oracle Database Standard Edition license.
  ```

  In this case, **Compute auto scaling** is disabled and the OCPU count is set to 8.

- If you switch to License Type: Bring your own license (BYOL) Standard Edition when the base OCPU count is below 8 and above 2, with **Compute auto scaling** enabled, you see the following message:
  `Your compute auto scaling maximum will be set to 8 OCPUs, under an Oracle Database Standard Edition license.`

  In this case, the maximum number of OCPUs with **Compute auto scaling** enabled is 8.

Update to ECPU Billing Model on Autonomous Database

Describes how to update an Autonomous Database instance from the OCPU billing model to the ECPU billing model.

> **Note:**
>
> If you update an Autonomous Database instance to use the ECPU compute model, you cannot revert back to the OCPU compute model.

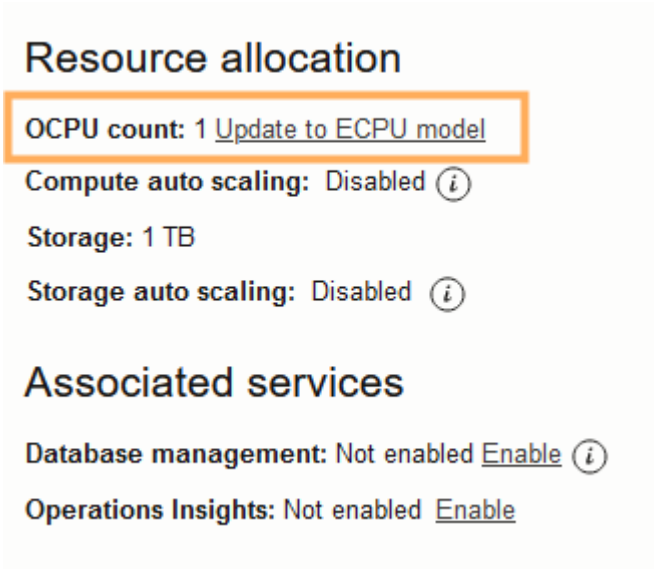
See Compute Models in Autonomous Database for details on the Autonomous Database compute models.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To update to the ECPU compute model:

1. On the **Details** page, under **Resource allocation**, in the **OCPU count field** click **Update to ECPU model**.



2. In the **Update billing model** area.

- **Compute**: For the compute options **ECPU count** and **Compute auto scaling**, the values are read only and you must accept the defaults. You can change the values after you update to ECPU billing model.

  See Use Auto Scaling for more information.

- **Backup retention**: The default backup retention is 60 days.

  To change the default specify a value for **Automatic backup retention period in days**.

  See Edit Automatic Backup Retention Period on Autonomous Database for more information.

3. Click **Save changes**.

While the system applies the changes, the lifecycle state changes to **Updating**.

When the operation completes the lifecycle state shows **Available** and on the details page under **Resource allocation**, the fields show **ECPU count** and **Compute auto scaling**.

Notes for Updating to the ECPU billing model:

- Your current storage selection is retained. You may scale up or down your storage after updating the billing model. See Add CPU or Storage Resources or Enable Auto Scaling for more information.
- Updating to the ECPU compute model impacts the cost of your compute, storage, and backups. See Compute Models in Autonomous Database for details on the ECPU compute model.
- Peer databases connected to an Autonomous Database instance that is updated to the ECPU compute model are also updated to the ECPU compute model.

  See Use Standby Databases with Autonomous Data Guard for Disaster Recovery and Use Backup-Based Disaster Recovery for more information on the Autonomous Database disaster recovery options.
- Any refreshable clones associated with an Autonomous Database instance that you update to use the ECPU compute model do not change their compute model (the associated refreshable clones continue to use the OCPU compute model). On a refreshable clone, you can update the instance to use the ECPU compute model. See Use Refreshable Clones with Autonomous Database for more information.

Update Always Free Instance to Paid with Autonomous Database

Describes how to update your instance to paid from free with Autonomous Database.

You can upgrade from an Always Free Autonomous Database account to a paid account at any time. If you have a free Oracle Cloud Infrastructure account this is a two step process:

First, upgrade to a paid Oracle Cloud Infrastructure account.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Always Free Autonomous Database instance from the links under the **Display Name** column.

> **Note:**
>
> Promotion of Always Free to a paid Autonomous Database is supported only if the database version for the Always Free Autonomous Database is Oracle Database 19c.

Upgrade your Always Free Autonomous Database instance as follows:

1. On the **Details** page, from the **More actions** drop-down list, select **Update instance to paid**.
2. On the **Update Instance** page confirm and proceed with the upgrade.

When you upgrade from an Always Free instance to a paid instance you get a paid instance of the same workload type with the minimum CPU and minimum storage available for that workload type. After you upgrade you can scale up CPU and storage resources to fit your needs.

For example:

- **Data Warehouse** workload type: When you upgrade to paid from an Always Free instance, the upgrade process provides a paid Autonomous Database instance with workload type **Data Warehouse**, with 2 ECPUs and 1 TB of database storage (if your instance is not enabled to use ECPUs you get 1 OCPU and 1 TB of database storage).
- **Transaction Processing** workload type: When you upgrade to paid from an Always Free instance, the upgrade process provides a paid Autonomous Database instance with workload type **Transaction Processing**, with 2 ECPUs and 1 TB of database storage (if your instance is not enabled to use ECPUs you get 1 OCPU and 1 TB of database storage).

Upgrade Autonomous JSON Database to Autonomous Transaction Processing

You can promote an Autonomous JSON Database to an Autonomous Transaction Processing database at any time.

An Autonomous JSON Database is the same as an Autonomous Transaction Processing database, except that an Autonomous JSON Database is limited in these respects:

- You can store only up to 20 GB of data other than JSON document collections.[1]

  (All Autonomous Databases, including , limit the storage of JSON data to 128 TB.)
- Collections cannot be **heterogeneous**. That is, they can only contain JSON documents. For example, you cannot have a collection of image documents or a collection that contains both JSON documents and image documents.

These limitations are appropriate if your use is primarily development of applications that use JSON documents. If you have a greater need to use data other than JSON data then follow these steps to promote your database to an Autonomous Transaction Processing database:

1. Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
2. From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database**, and then click Autonomous JSON Database.

[1] You can subscribe to information event AJDNonJsonStorageExceeded, to be informed when the 20 GB limit is exceeded. See About Information Events on Autonomous Database.

3. Choose your JSON Database from those listed in the compartment, by clicking its name in column **Display name**.
4. Do one of the following:
   - From the **More actions** drop-down list, select **Change workload type**.
   - In tab Autonomous Database information, under heading **General information**, item **Workload type**, click **Edit**.
5. Click **Convert** to confirm that you want to convert this database to Autonomous Transaction Processing.
6. If you were using the refreshable clone feature with your Autonomous JSON Database then re-create the clone after promotion to Autonomous Transaction Processing. See Using Refreshable Clones with Autonomous Database.

See Autonomous Database Billing Summary for more information.

Upgrade APEX Service to Autonomous Transaction Processing

Easily upgrade APEX Service to Autonomous Transaction Processing.

If you want to expand your capabilities to include the full features of Oracle Autonomous Database with APEX, you can easily upgrade an APEX Service instance to a full Oracle Autonomous Transaction Processing instance. You can upgrade from either the APEX Instances Details page or the Autonomous Database Details page.

To upgrade to Oracle Autonomous Transaction Processing:

1. From the Oracle Cloud Infrastructure left navigation list, select **Developer Services** and then click **APEX Application Development** or **APEX Instances**.

   The APEX Instances page appears.
2. Under **Filters**, narrow the display by selecting **APEX** from the Type list.
3. On the APEX Instances page, click the APEX instance name.

   The APEX Instance Details page appears.
4. You can upgrade in two ways:
   - On the APEX Instance Details page:
     - Under **APEX instance information**, find **Database type** and click **Edit**.
   - Navigate to the Autonomous Database Details page:
     - Under **APEX instance information**, click the Database name.
     - Under **General information**, find **Workload type** and click **Edit**.

     A dialog appears.
5. Review the information in the dialog and then click **Convert** to convert the database to Autonomous Transaction Processing.

See Autonomous Database Billing Summary for more information.

Cloning and Moving an Autonomous Database

Autonomous Database provides cloning where you can choose to create a full clone, create a metadata clone, or create a refreshable clone. You can also move a database to a different compartment.

- About Cloning on Autonomous Database
  When you create a clone for an Autonomous Database instance, you have the option to select the clone type: a full clone, a metadata clone, or a refreshable clone:
- Prerequisites for Cloning
  Describes prerequisites for cloning an Autonomous Database instance.
- Clone an Autonomous Database Instance
  Shows you the steps to clone an Autonomous Database from the Oracle Cloud Infrastructure Console.
- Clone an Autonomous Database from a Backup
  Shows the options to select a backup as the clone source for cloning Autonomous Database.
- Cross Tenancy and Cross-Region Cloning
  You can clone an Autonomous Database instance from one tenancy, source tenancy, to a different tenancy (destination tenancy).
- Clone Autonomous Database to Change Workload Type
  When you create an Autonomous Database clone, by default the clone is of the same workload type. Optionally, you can select that the clone is of a different workload type than the source database.
- Notes for Cloning Autonomous Database
  Provides information about the cloning operation and the resulting cloned database.
- Move an Autonomous Database to a Different Compartment
  Shows you the steps to move a database to a different Oracle Cloud Infrastructure compartment.

About Cloning on Autonomous Database

When you create a clone for an Autonomous Database instance, you have the option to select the clone type: a full clone, a metadata clone, or a refreshable clone:

- **Full Clone**: creates a new database with the source database's data and metadata.
- **Refreshable Clone**: creates a read-only full clone that can be easily refreshed with the data from the source database.

  See Use Refreshable Clones with Autonomous Database for more information.
- **Metadata Clone**: creates a new database that includes all of the source database schema metadata, but not the source database data.

For a Full Clone or a Metadata Clone, you have the option to select the clone source:

- **Clone from a database instance**: This creates a clone of a running database.
- **Clone from a backup**: This creates a clone when you select a backup from a list of backups, or when you enter a backup point-in-time to clone.
- **Clone from the latest backup**: This creates a clone where Autonomous Database uses the most recent backup data that is available to create the clone.

> **Note:**
>
> Depending on the cloning options you choose, cloning an Autonomous Database instance copies your database files to a new instance. Depending on the size of your database and where you are cloning your database, expect the cloning operation to take significantly longer than provisioning a new Autonomous Database instance. There is no downtime associated with cloning and the cloning operation has no impact on applications running on the source.

Additional Cloning Options

There are several additional cloning options that allow you to clone to a different workload type, a different region, or to a different tenancy.

Cloning Option	Description
Cross-Region Cloning	You can clone a database where the source and the cloned database are in different regions (cross-region cloning). This allows you to clone an existing database or clone a backup to create a database in a different region.
Clone to Change the Workload Type	You can clone a database and select a different workload type for the cloned database. See Clone Autonomous Database to Change Workload Type for more information.
Source Unavailable Cloning	You can clone a database when the source Autonomous Database instance is in the unavailable state. This allows you to create a new database from a backup and use the new database in place of the unavailable database.
Preview Release Cloning	When an upcoming release is available for preview with Autonomous Database, you can clone your existing database to use the preview version. A preview version is only available when there is an upcoming major version to release. When no preview version is available, cloning does not provide this option.
Cross Tenancy Cloning	You can clone a database where the source and the cloned database are in different tenancies (cross-tenancy cloning). This allows you to clone an existing database or clone a backup to create a database in a different tenancy, where the clone is either in a different tenancy in the same region or in a different tenancy in a different region. **Note:** The cross tenancy cloning option is only available using the CLI or the Autonomous Database REST APIs. This option is not available using the Oracle Cloud Infrastructure Console. See Cross Tenancy and Cross-Region Cloning for more information.
Move to a Different Compartment	You can also move an Autonomous Database to a different Oracle Cloud Infrastructure Compartment. See Move an Autonomous Database to a Different Compartment for more information.

Prerequisites for Cloning

Describes prerequisites for cloning an Autonomous Database instance.

To clone an Autonomous Database instance you must define the required access using OCI Identity and Access Management policy statements written by an administrator, whether you're using the Console, the REST API, the CLI, or another tool.

The following policies allow you to create a clone:

```
Allow group Group_Name to read autonomous-databases in compartment
Compartment_Name
      where target.id = 'oc1.autonomousdatabase.oc1..unique_ID'


Allow group Group_Name to manage autonomous-databases in compartment
Compartment_Name
```

The where clause is optional and provides a more fine grained way to grant access to a specific database.

You can limit cloning permissions so that the group can only clone Autonomous Databases but cannot create Autonomous Databases, or further limit permission to only create a particular type of clone: Full Clone, Metadata Clone, or Refreshable Clone. See IAM Permissions and API Operations for Autonomous Database for more information and examples.

This shows the policies for cloning within a tenancy. See Prerequisites for Cross Tenancy Cloning for cross tenancy cloning policies.

See IAM Policies for Autonomous Database and Getting Started with Policies for more information.

Clone an Autonomous Database Instance

Shows you the steps to clone an Autonomous Database from the Oracle Cloud Infrastructure Console.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- Choose your region. See Switching Regions for information on switching regions and working in multiple regions.
- Choose your **Compartment**. See Compartments for information on using and managing compartments.
- Select an Autonomous Database instance from the list in your compartment.

To clone the Autonomous Database instance:

1. On the **Autonomous Database Details** page, from the **More actions** drop-down list, select **Create clone**.

2. On the **Create Autonomous Database clone** page, choose the clone type from the choices:
   - **Full clone**: creates a new database with the source database's data and metadata.
   - **Refreshable clone**: creates a read-only full clone that can be easily refreshed with the source database's data.
     See Use Refreshable Clones with Autonomous Database for more information.
   - **Metadata clone**: creates a new database with the source database's metadata without the data.
3. In the **Clone source** area, select one of:
   - **Clone from database instance**: This creates a clone from a running database.
   - **Clone from a backup**: This creates a database clone from a selected backup, from a point-in-time timestamp, or from the latest backup data that is available. See Clone an Autonomous Database from a Backup for more information.
4. Provide basic information for the Autonomous Database clone.
   - **Choose your preferred region**: from the list, select the region where you want to create the clone.

     Note: the list only shows the regions that you are subscribed to.

     For the clone type **Clone from database instance** as the clone source, when you choose another region other than the current region for your clone target, using either the Oracle Cloud Infrastructure CLI or Terraform you can only perform such a cross-region clone from the remote region. That is, call the create clone API from the remote region to which you want to clone, with the source database OCID as that of the source that you want to clone from.
   - **Create in Compartment**: See Compartments for information on using and managing compartments.

     Note: the list only shows the compartments that you are subscribed to.
   - **Display name**: Specify a user-friendly description or other information that helps you easily identify the resource.

     You can use the name provided, of the form: **Clone-of-***DBname* or change this to the name you want to use to identify the database. The supplied *DBname* is the name of the source database that you are cloning.
   - **Database name**: Specify the database name; it must consist of letters and numbers only. The maximum length is 30 characters. The same database name cannot be used for multiple Autonomous Databases in the same tenancy in the same region.

     The default database name is a generated 16-character string.
5. Select a value for **Choose a workload type** from the following options.
   - **Data Warehouse**: This creates an Autonomous Data Warehouse type clone.
   - **Transaction Processing**: This creates an Autonomous Transaction Processing type clone.
   - **JSON**: This creates an Autonomous JSON Database type clone.
   - **APEX**: This creates an APEX type clone.

> **Note:**
>
> The unavailable cloning options are grayed out. See Clone Autonomous Database to Change Workload Type for more information on cross workload cloning.

6. Configure the database (ECPU compute model)
   - **Always Free**: Select to show Always Free options.

     You can only create a free instance in the tenancy's Home region.
   - **Choose database version**: Select the database version. The available database version is 19c.

     With Always Free selected, the available database versions are: Oracle Database 19c and Oracle Database 23ai. With Always Free selected, cloning is only allowed from Oracle Database 23ai to Oracle Database 23ai.
   - **ECPU count**: Specify the number of CPUs for your database. The minimum value for the number of ECPUs is 2.

     Your license type determines the **ECPU count** maximum. For example, if your license type is Bring your own license (BYOL) with Oracle Database Standard Edition (SE), the **ECPU count** maximum is 32.
   - **Compute auto scaling**: By default compute auto scaling is enabled to allow the system to automatically use up to three times more CPU and IO resources to meet workload demand. If you do not want to use compute auto scaling then deselect this option.

     See Use Auto Scaling for more information.
   - **Storage**: Specify the storage you wish to make available to your database. Depending on your workload type you have these options:
     - **Data Warehouse**: Specify your storage in Terabytes (TB).

       For a Full Clone, the minimum storage that you can specify is the source database's actual used space rounded to the next TB.
     - **Transaction Processing**: Specify your storage in Gigabytes (GB) or Terabytes (TB). Enter the size in the **Storage** field. Select **GB** or **TB** for the **Storage unit size**.

       For a Full Clone, the minimum storage that you can specify is the source database's actual used space rounded to the next GB.

     By default, the IO capacity of your database depends on the number of ECPUs you provision. When you provision 384 TB of storage, your database is entitled to the full IO capacity of the Exadata infrastructure, independent of the number of ECPUs you provision.

     If you want to provision more than 384 TB of storage, file a Service Request at Oracle Cloud Support.
   - **Storage auto scaling**: By default storage auto scaling is disabled. Select if you want to enable storage auto scaling to allow the system to automatically expand to use up to three times more storage.

     See Use Auto Scaling for more information.
   - **Show advanced options**: Click to show the compute model options or if you want to create or join an elastic pool:

- **Enable elastic pool**:

  See Create or Join an Elastic Pool While Provisioning or Cloning an Instance for more information.

- **Compute model**: Shows the selected compute model.

7. Backup retention

   By default the automatic backup retention is 60 days.

   Automatic backup retention period in days: You have the option to select an automatic backup retention period, in a range from 1 to 60 days. You can restore and recover your database to any point-in-time in this retention period.

   This option is not available with the OCPU compute model.

   See About Backup and Recovery on Autonomous Database for more information.

8. Create administrator credentials.

   - **Username** This is a read-only field.
   - **Password** Set the password for the Autonomous Database Admin user. The password must meet the strong password complexity criteria based on Oracle Cloud security standards. For more information on the password complexity rules see Create Users on Autonomous Database - Connecting with a Client Tool.
   - **Confirm password** Specify a value to confirm the password.

9. Choose network access

   > **Note:**
   >
   > After you clone your Autonomous Database you can change the network access option you select for the cloned instance.

   - **Secure access from everywhere**

     By default all secure connections are allowed from everywhere.

   - **Secure access from allowed IPs and VCNs only**

     This option restricts connections to the database according to the access control rules (ACLs) you specify. To add multiple ACLs for the Autonomous Database, click **Add access control rule**.

     See Configure Access Control Lists When You Provision or Clone an Instance for more information.

   - **Private endpoint access only**

     This option assigns a private endpoint, private IP, and hostname to your database. Specifying this option allows traffic only from the VCN you specify; access to the database from all public IPs or VCNs is blocked. This allows you to define security rules, ingress/egress, at the Network Security Group (NSG) level and to control traffic to your Autonomous Database.

     See Configure Private Endpoints When You Provision or Clone an Instance for more information.

10. Choose license and Oracle Database edition

    - **License included**

The default license type is license included. With this option, when you create a database by cloning you subscribe to new database software licenses and the database cloud service.

- **Switch to Bring your own license (BYOL)**

  Select this to show more license options. Select this if your organization already owns Oracle Database software licenses and you want to bring your existing database software licenses to the database cloud service. See Cloud pricing for information on Bring your own license (BYOL) and other licensing options for Oracle Cloud Infrastructure cloud service pricing.

  When you select to switch to **Bring your own license (BYOL)**, you also select an Oracle Database edition. The Oracle Database edition you select is based on the license you bring to Autonomous Database and changes the maximum value that you can select for the **ECPU count**. The choices are:

  – **Oracle Database Enterprise Edition (EE)**: For this license type the maximum allowed value for **ECPU count** is 512, however you may contact your Oracle account team to request more ECPUs. With compute auto scaling enabled you can use up to **ECPU count** x 3 ECPUs. For example, if you set the **ECPU count** to 512, you can use up to 1,536 ECPUs.

  – **Oracle Database Standard Edition (SE)**: For this license type the maximum allowed value for **ECPU count** is 32. With compute auto scaling enabled you can use up to **ECPU count** x 3 ECPUs. This license restricts the number of ECPUs you can use to a maximum of 32 ECPUs, with or without compute auto scaling enabled.

    See Use Auto Scaling for more information.

11. (Optional) Provide contacts for operational notifications and announcements

    Click **Add contact** and in the **Contact email** field, enter a valid email address. If the database you are cloning has a customer contact list, the list is copied. To enter multiple **Contact email** addresses, repeat the process to add up to 10 customer contact emails.

    See View and Manage Customer Contacts for Operational Issues and Announcements for more information.

12. (Optional) Click Show advanced options to select advanced options.

    - **Encryption key**
      Encrypt using an Oracle-managed key: By default Autonomous Database uses Oracle-managed encryption keys. Using Oracle-managed keys, Autonomous Database creates and manages the encryption keys that protect your data and Oracle handles rotation of the TDE master key.

      Encrypt using a customer-managed key in this tenancy: If you this option, a master encryption key from a Oracle Cloud Infrastructure Vault in the same tenancy is used to generate the TDE master key on Autonomous Database.

      Encrypt using a customer-managed key located in a remote tenancy: If you this option, a master encryption key in the Oracle Cloud Infrastructure Vault located in a remote tenancy is used to generate the TDE master key on Autonomous Database.

      See Use Customer-Managed Encryption Keys on Autonomous Database for more information.

    - **Maintenance**
      Patch level By default the patch level is the patch level of the source database. Select **Early** to configure the instance with the early patch level. When cloning a source database with **Early** patch level, you can only choose the **Early** patch level for your clone.

See Set the Patch Level for more information.

- **Management**
  Shows the character set and national character set for your database.

  See Choose a Character Set for Autonomous Database for more information.

- **Tools**
  If you want to view or customize the tools configuration, select the tools tab.

  See Configure Autonomous Database Built-in Tools when you Provision or Clone an Instance for more information.

- **Tags**
  If you want to use Tags, enter the **Tag key** and **Tag value**. Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values which can be attached to resources.

  See Tagging Overview for more information.

13. Click **Create Autonomous Database clone**.

On the Oracle Cloud Infrastructure console the **State** shows Provisioning... until the new database is available.

If you created a cross-region clone, a new tab shows the newly provisioned clone's Oracle Cloud Infrastructure Console and the region shown is the region you selected when you created the clone.

See Notes for Cloning Autonomous Database for additional information on cloning.

See Cloning an Autonomous Database for information on using the API.

Clone an Autonomous Database from a Backup

Shows the options to select a backup as the clone source for cloning Autonomous Database.

Perform the following prerequisite steps as necessary:

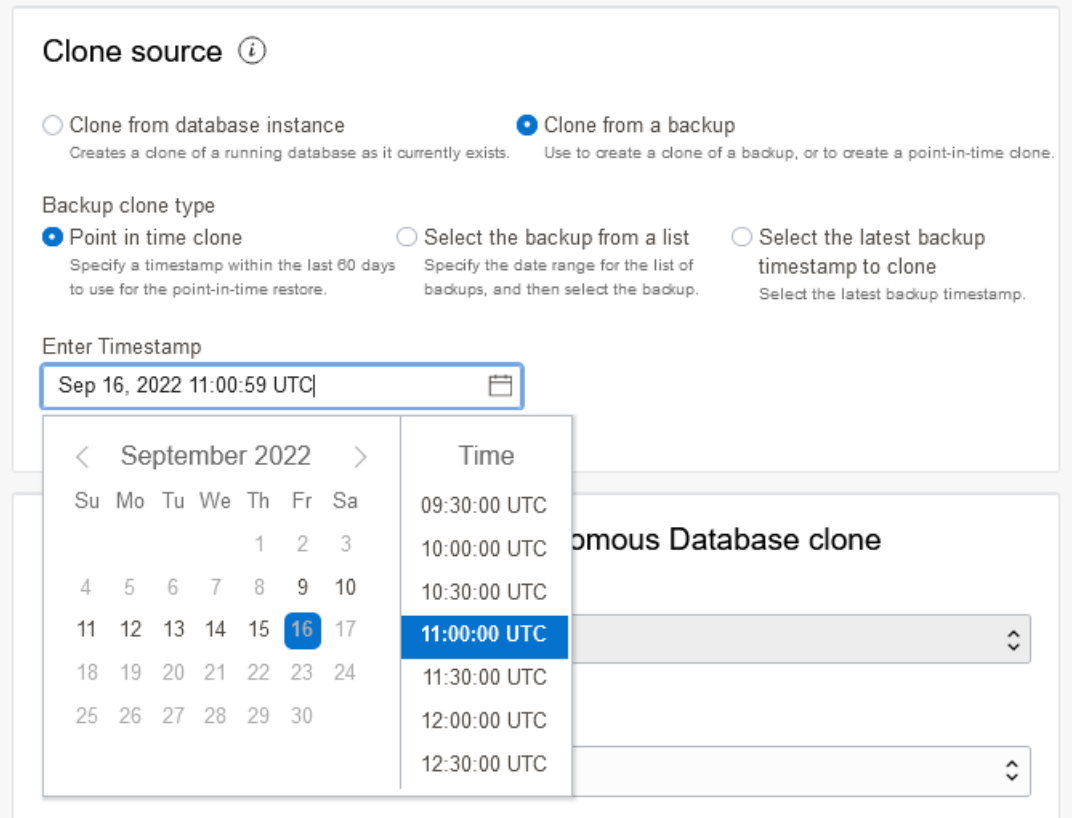
- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.

1. Choose your region. See Switching Regions for information on switching regions and working in multiple regions.
2. Choose your **Compartment**. See Compartments for information on using and managing compartments.
3. Select an Autonomous Database instance from the list in your compartment.
4. On the **Details** page, from the **More actions** drop-down list, select **Create clone**.
5. On the **Create Autonomous Database clone** page, choose the clone type from the choices:
   - **Full clone**: creates a new database with the source database's data and metadata.
   - **Refreshable clone**: creates a read-only full clone that can be easily refreshed with the source database's data.
     See Use Refreshable Clones with Autonomous Database for more information.

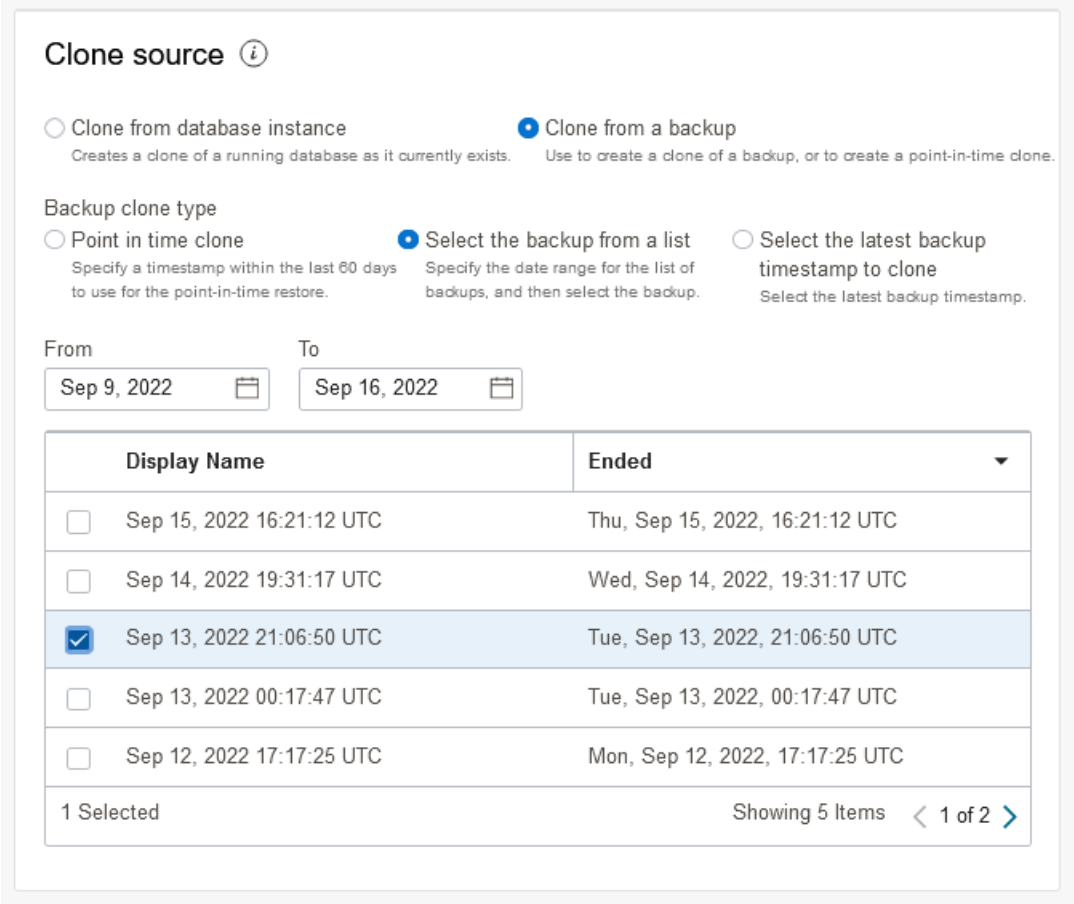
- **Metadata clone**: creates a new database with the source database's metadata without the data.

6. In the **Configure clone source** area, select the **Clone source** option:
   - **Clone from database instance**: This creates a clone from a running database. See Clone an Autonomous Database Instance for details and steps with this selection.
   - **Clone from a backup**: This selection creates a database clone using a backup. Select this option.
7. In the **Configure clone source** area, select the **Backup clone type**:
   - **Point in time clone**: Enter a timestamp to clone in the **Enter Timestamp** field.
   - **Select the backup from a list**: Enter **From** and **To** dates to narrow the list of backups then select a backup to use for the clone source.
   - **Latest backup timestamp**: See Clone an Autonomous Database from Latest Backup for information on this clone option.

Point in time clone



Select the backup from a list



8. Provide basic information for the Autonomous Database.
   - **Choose your preferred region**: from the list, select the region where you want to create the clone.

     Note: the list only shows the regions that you are subscribed to.

     For the clone type **Clone from a backup** as the clone source, when you choose another region other than the current region for your clone target, using either the Oracle Cloud Infrastructure CLI or Terraform you can only perform such a cross-region clone from the remote region. That is, call the create clone API from the remote region to which you want to clone, with the source database OCID as that of the source that you want to clone from.
   - **Create in Compartment**: See Compartments for information on using and managing compartments.
   - **Display name**: Specify a user-friendly description or other information that helps you easily identify the resource.

     You can use the name provided, of the form: **Clone-of-***DBname* or change this to the name you want to use to identify the database. The supplied *DBname* is the name of the source database that you are cloning.
   - **Database Name**: Specify the database name; it must consist of letters and numbers only. The maximum length is 30 characters. The same database name cannot be used for multiple Autonomous Databases in the same tenancy in the same region.

     The default database name is a generated 16-character string.

9. Select a value for **Choose a workload type** from the following options.
   - **Data Warehouse**: This creates an Autonomous Data Warehouse type clone.
   - **Transaction Processing**: This creates an Autonomous Transaction Processing type clone.
   - **JSON**: This creates an Autonomous JSON Database type clone.
   - **APEX**: This creates an APEX type clone.

   > **Note:**
   >
   > The unavailable cloning options are grayed out. See Clone Autonomous Database to Change Workload Type for more information on cross workload cloning.

10. Configure the database (ECPU compute model)
    - **Always Free**: Select to show Always Free options.

      You can only create a free instance in the tenancy's Home region.
    - **Choose database version**: Select the database version. The available database version is 19c.

      With Always Free selected, the available database versions are: Oracle Database 19c and Oracle Database 23ai. With Always Free selected, cloning is only allowed from Oracle Database 23ai to Oracle Database 23ai.
    - **ECPU count**: Specify the number of CPUs for your database. The minimum value for the number of ECPUs is 2.

      Your license type determines the **ECPU count** maximum. For example, if your license type is Bring your own license (BYOL) with Oracle Database Standard Edition (SE), the **ECPU count** maximum is 32.
    - **Compute auto scaling**: By default compute auto scaling is enabled to allow the system to automatically use up to three times more CPU and IO resources to meet workload demand. If you do not want to use compute auto scaling then deselect this option.

      See Use Auto Scaling for more information.
    - **Storage**: Specify the storage you wish to make available to your database. Depending on your workload type you have these options:
      - **Data Warehouse**: Specify your storage in Terabytes (TB).

        For a Full Clone, the minimum storage that you can specify is the source database's actual used space rounded to the next TB.
      - **Transaction Processing**: Specify your storage in Gigabytes (GB) or Terabytes (TB). Enter the size in the **Storage** field. Select **GB** or **TB** for the **Storage unit size**.

        For a Full Clone, the minimum storage that you can specify is the source database's actual used space rounded to the next GB.

      By default, the IO capacity of your database depends on the number of ECPUs you provision. When you provision 384 TB of storage, your database is entitled to the full IO capacity of the Exadata infrastructure, independent of the number of ECPUs you provision.

If you want to provision more than 384 TB of storage, file a Service Request at Oracle Cloud Support.

- **Storage auto scaling** By default storage auto scaling is disabled. Select if you want to enable storage auto scaling to allow the system to automatically expand to use up to three times more storage.

  See Use Auto Scaling for more information.

- **Show advanced options**: Click to show the compute model option:
  - **Enable elastic pool**:

    See Create or Join an Elastic Pool While Provisioning or Cloning an Instance for more information.
  - **Compute model**: Shows the selected compute model.

11. Backup retention

    By default the automatic backup retention is 60 days.

    Automatic backup retention period in days You have the option to select the automatic backup retention period, in a range from 1 to 60 days. You can restore and recover your database to any point-in-time in this retention period.

    This option is not available with the OCPU compute model.

    See About Backup and Recovery on Autonomous Database for more information.

12. Create administrator credentials.
    - **Username** This is a read-only field.
    - **Password** Set the password for the Autonomous Database Admin user. The password must meet the strong password complexity criteria based on Oracle Cloud security standards. For more information on the password complexity rules see Create Users on Autonomous Database - Connecting with a Client Tool.
    - **Confirm password** Specify a value to confirm the password.
13. Choose network access

    > **Note:**
    >
    > After you clone your Autonomous Database you can change the network access option you select for the cloned instance.

    - **Secure access from everywhere**

      By default all secure connections are allowed from everywhere.
    - **Secure access from allowed IPs and VCNs only**

      This option restricts connections to the database according to the access control rules (ACLs) you specify. To add multiple ACLs for the Autonomous Database, select this option and click **Add access control rule**.

      See Configure Access Control Lists When You Provision or Clone an Instance for more information.
    - **Private endpoint access only**

      This option assigns a private endpoint, private IP, and hostname to your database. Specifying this option allows traffic only from the VCN you specify; access to the

database from all public IPs or VCNs is blocked. This allows you to define security rules, ingress/egress, at the Network Security Group (NSG) level and to control traffic to your Autonomous Database.

See Configure Private Endpoints When You Provision or Clone an Instance for more information.

14. Choose license and Oracle Database edition

    - **License included**

      The default license type is license included. With this option, when you create a database by cloning you subscribe to new database software licenses and the database cloud service.

    - **Switch to Bring your own license (BYOL)**

      Select this to show more license options. Select this if your organization already owns Oracle Database software licenses and you want to bring your existing database software licenses to the database cloud service. See Cloud pricing for information on Bring your own license (BYOL) and other licensing options for Oracle Cloud Infrastructure cloud service pricing.

      When you select to switch to **Bring your own license (BYOL)**, you also select an Oracle Database edition. The Oracle Database edition you select is based on the license you bring to Autonomous Database and changes the maximum value that you can select for the **ECPU count**. The choices are:

      – **Oracle Database Enterprise Edition (EE)**: For this license type the maximum allowed value for **ECPU count** is 512, however you may contact your Oracle account team to request more ECPUs. With compute auto scaling enabled you can use up to **ECPU count** x 3 ECPUs. For example, if you set the **ECPU count** to 512, you can use up to 1,536 ECPUs.

      – **Oracle Database Standard Edition (SE)**: For this license type the maximum allowed value for **ECPU count** is 32. With compute auto scaling enabled you can use up to **ECPU count** x 3 ECPUs. This license restricts the number of ECPUs you can use to a maximum of 32 ECPUs, with or without compute auto scaling enabled.

        See Use Auto Scaling for more information.

15. (Optional) Provide contacts for operational notifications and announcements

    Click **Add contact** and in the **Contact email** field, enter a valid email address. If the database you are cloning has a customer contact list, the list is copied. To enter multiple **Contact email** addresses, repeat the process to add up to 10 customer contact emails.

    See View and Manage Customer Contacts for Operational Issues and Announcements for more information.

16. (Optional) Click Show advanced options to select advanced options.

    - **Encryption Key**
      Encrypt using an Oracle-managed key: By default Autonomous Database uses Oracle-managed encryption keys. Using Oracle-managed keys, Autonomous Database creates and manages the encryption keys that protect your data and Oracle handles rotation of the TDE master key.

      Encrypt using a customer-managed key in this tenancy: If you this option, a master encryption key from a Oracle Cloud Infrastructure Vault in the same tenancy is used to generate the TDE master key on Autonomous Database.

Encrypt using a customer-managed key located in a remote tenancy: If you this option, a master encryption key in the Oracle Cloud Infrastructure Vault located in a remote tenancy is used to generate the TDE master key on Autonomous Database.

See Use Customer-Managed Encryption Keys on Autonomous Database for more information.

- **Maintenance**
  Patch level By default the patch level is the patch level of the source database. Select **Early** to configure the instance with the early patch level. When cloning a source database with **Early** patch level, you can only choose the **Early** patch level for your clone.

  See Set the Patch Level for more information.

- **Management**
  Shows the character set and national character set for your database.

  See Choose a Character Set for Autonomous Database for more information.

- **Tools**
  If you want to view or customize the tools configuration, select the tools tab.

  See Configure Autonomous Database Built-in Tools when you Provision or Clone an Instance for more information.

- **Tags**
  If you want to use Tags, enter the **Tag key** and **Tag value**. Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values which can be attached to resources.

  See Tagging Overview for more information.

17. Click **Create Autonomous Database Clone**.

On the Oracle Cloud Infrastructure console the **State** shows Provisioning... until the new database is available.

> **Notes:**
>
> - If there is an ongoing clone from backup operation on a source database, you cannot initiate a new clone operation on the same backup being cloned until the ongoing operation completes. Thus, you cannot clone from backup twice concurrently from a specific backup (for example, a specific timestamp or a specific selected backup from the list of backups).
> - For external tables, partitioned external tables, and the external partitions of hybrid partitioned tables a backup does not include the external files that reside on your Object Store. Thus, for the clone from backup operation, it is your responsibility to backup, and restore if necessary, the external files associated with external tables, external partitioned tables, or the external files for a hybrid partitioned table
> - With clone from backup, the Oracle Machine Learning workspaces, projects, and notebooks of the source database are not cloned to the new database.

See Notes for Cloning Autonomous Database for additional information on cloning.

See Cloning an Autonomous Database for information on using the API.

- Clone an Autonomous Database from Latest Backup
  When you choose to clone from latest backup, this selects the latest backup as the clone source. You can choose this option if a database becomes unavailable or for any reason when you want to create a clone based on the most recent backup.

Clone an Autonomous Database from Latest Backup

When you choose to clone from latest backup, this selects the latest backup as the clone source. You can choose this option if a database becomes unavailable or for any reason when you want to create a clone based on the most recent backup.

Follow the steps in Clone an Autonomous Database from a Backup and in the Clone source area, select **Clone from a backup** and **Select the latest backup timestamp to clone** to create a clone that recovers the most recent backup data that is available for the Autonomous Database instance (the clone source) from the database backup and logs backup.

After Autonomous Database has finished provisioning the clone, query the view `dba_pdbs` to see the `last_recover_time` and `last_recover_scn` columns. These columns provide the saved timestamp and the saved SCN of the data from which the cloned database was created.

For example:

```
SELECT last_recover_time, last_recover_scn FROM dba_pdbs;
```

Cross Tenancy and Cross-Region Cloning

You can clone an Autonomous Database instance from one tenancy, source tenancy, to a different tenancy (destination tenancy).

- About Cross Tenancy Cloning
  When you create a cross tenancy clone you can either select that the clone is created in the same region as the source tenancy or in a different region than the source tenancy (cross-region).
- Prerequisites for Cross Tenancy Cloning
  Describes prerequisites for creating a cross tenancy clone where the source database is in one tenancy and the cloned database is in a different tenancy.
- Create a Cross Tenancy or Cross-Region Clone
  Shows the steps to create a cross tenancy clone when the source database and the cloned database are in the same region, or when the source database and the cloned database are in different regions (cross-region).
- Create a Cross Tenancy Clone from a Backup
  Shows the steps to create a cross tenancy clone from a backup.

About Cross Tenancy Cloning

When you create a cross tenancy clone you can either select that the clone is created in the same region as the source tenancy or in a different region than the source tenancy (cross-region).

> **Note:**
>
> The cross tenancy cloning option is only available using the CLI or the Autonomous Database REST APIs. This option is not available using the Oracle Cloud Infrastructure Console.

Note the following for cross tenancy cloning:

- All clone types are supported: the cloned database can be a Full clone, a Metadata clone, or a Refreshable clone.
- A clone can be created from a source Autonomous Database instance or from a backup (using the latest backup, a specified backup, or by selecting a long-term backup).
- The source Autonomous Database instance can use either the ECPU or OCPU compute model. Depending on your workload type, you can clone from a source that uses the OCPU compute model to a clone that uses the ECPU compute model (this is allowed for the Data Warehouse and the Transaction Processing workload types).
- The cloned database can be in the same region or in a different region (cross-region).
- The cross tenancy cloning option does not support cloning with customer managed keys on the source. See Manage Encryption Keys on Autonomous Database for more information on customer managed keys.

Prerequisites for Cross Tenancy Cloning

Describes prerequisites for creating a cross tenancy clone where the source database is in one tenancy and the cloned database is in a different tenancy.

You must run the commands to create a cross tenancy clone on the destination tenancy. Before you create a cross tenancy clone you need to define OCI Identity and Access Management groups and policies on the source tenancy, the tenancy that contains the instance you are cloning, and on the destination tenancy. The groups and policies you define allow you to run commands to create the clone on the destination tenancy and allow the destination tenancy to contact the source tenancy where the source Autonomous Database instance resides.

The OCI Identity and Access Management groups and policies you add support the following:

- A member of a group in the source tenancy allows a group in the destination tenancy to access (read) the source Autonomous Database instance on the source tenancy.

  You do not need to allow other actions on the source Autonomous Database instance (for example, start, stop terminate, or any write operations).
- A member of a group in the destination tenancy is allowed to create a clone in the destination tenancy using the Autonomous Database instance in the source tenancy as the clone source.

  On the destination tenancy you also add a policy that allows a group to manage the Autonomous Database instance on the source tenancy. For example, this policy allows the group to create the clone database, and allows a refreshable clone to run commands that contact the source tenancy, such as **Refresh** and **Disconnect**.

To create a cross tenancy clone use OCI Identity and Access Management to create the required groups and to define the policies that authorize cross tenancy cloning:

1. Create a group on the destination tenancy that contains the user that will be allowed to create a clone.

a. On the destination tenancy, in the Oracle Cloud Infrastructure Console click **Identity & Security**.

b. Under **Identity** click **Domains** and select an identity domain (or create a new identity domain).

c. Under **Identity domain**, click **Groups**.

d. To add a group, click **Create group**.

e. On the Create group page, enter a Name and a Description.

   For example, enter the Name: **DestinationGroup**.

f. On the Create group page, click **Create**.

g. Click **Create** to save the group.

h. On the **Group** page, click **Assign user to groups** and select the users you want to add to the group.

i. Click **Add**.

j. On the **Group** page, from the **Group information** tab copy the OCID for use in Step 2.

2. On the source tenancy, define OCI Identity and Access Management policies for the source Autonomous Database instance.

   a. On the source tenancy, in the Oracle Cloud Infrastructure Console click **Identity & Security**.

   b. Under **Identity**, click **Policies**.

   c. To write a policy, click **Create Policy**.

   d. On the Create Policy page enter a Name and a Description.

   e. On the Create Policy page, select **Show manual editor**.

   f. In the policy builder, add policies so that the group in the destination tenancy is allowed to create a clone using an Autonomous Database instance on the source tenancy as the clone source.

      For example, define the following generic policies:

```
define tenancy DestinationTenancy as ocid1.tenancy.oc1..unique_ID
define group DestinationGroup as ocid1.group.region1..unique_ID
admit group DestinationGroup of tenancy DestinationTenancy to read
autonomous-database-family
       in compartment ocid1.compartment.region1..unique_ID
       where target.id = 'oc1.autonomousdatabase.oc1..unique_ID'
```

      This policy specifies the following:

      - Line 1: the OCID is the OCID of the destination tenancy. This is the tenancy where you are going to create the clone.
      - Line 2: the OCID is the OCID of the group to which the user who will create the clone belongs. This is the OCID you created in Step 1.
      - Line 3: The first OCID is the OCID of the compartment where the source database resides. The second OCID, after the `where` clause, is the OCID of the source Autonomous Database instance.

> **Note:**
>
> The where clause is optional and provides a more fine grained way to grant access to a specific database.

For example, set these policies on the source tenancy to allow cross tenancy cloning:

```
define tenancy DestinationTenancy as
ocid1.tenancy.oc1..aaa_example_rcyx2a
define group DestinationGroup as ocid1.group.oc1..aaa_example_6vctn6xsaq
admit group DestinationGroup of tenancy DestinationTenancy to read
autonomous-database-family in compartment
   ocid1.compartment.region1..bbb_example_rcyx2b where target.id =
'oc1.autonomousdatabase.oc1.aaaabbbbcccc'
```

This policy specifies a user in the `DestinationGroup` of the `DestinationTenancy` can read from a specific Autonomous Database instance in the specified compartment (on the source tenancy). To create a cross tenancy clone the policy only needs to allow read on the source Autonomous Database instance.

g. Click **Create** to save the policy.

3. Define policies on the destination tenancy.

   a. On the destination tenancy, in the Oracle Cloud Infrastructure Console click **Identity & Security**.

   b. Under **Identity**, click **Policies**.

   c. To write a policy, click **Create Policy**.

   d. On the Create Policy page enter a Name and a Description.

   e. On the Create Policy page, select **Show manual editor**.

   f. In the policy builder, add policies so that a group is endorsed to manage Autonomous Databases on the source tenancy.

   For example:

   ```
   Define tenancy SourceTenancy as ocid1.tenancy.oc1..unique_ID
   Endorse group DestinationGroup to manage autonomous-database-family in
   tenancy SourceTenancy
   ```

   This policy specifies the following:

   - Line 1: The OCID is the source tenancy OCID. This is the tenancy where the source Autonomous Database instance resides.
   - Line 2: Specifies that the DestinationGroup group can manage Autonomous Databases in the source tenancy.

   Notes for defining policies on the destination tenancy:

   - For the following policy:

     ```
     Endorse group DestinationGroup to manage autonomous-database-family in
     tenancy SourceTenancy
     ```

This policy allows the group `DestinationGroup` to create Autonomous Databases and Autonomous Database clones in the source tenancy. You can limit cloning permissions so that the group can only clone Autonomous Databases but cannot create Autonomous Databases, or further limit permission to only create a particular type of clone: Full Clone, Metadata Clone, or Refreshable Clone. See IAM Permissions and API Operations for Autonomous Database for more information and examples.

- If these polices are revoked, cross tenancy cloning is no longer allowed.

See Getting Started with Policies for more information.

Create a Cross Tenancy or Cross-Region Clone

Shows the steps to create a cross tenancy clone when the source database and the cloned database are in the same region, or when the source database and the cloned database are in different regions (cross-region).

These steps cover creating a Full clone or a Metadata clone. See Create a Cross Tenancy or Cross-Region Refreshable Clone for details on creating a cross tenancy refreshable clone.

> **Note:**
>
> The cross tenancy cloning option is only available using the CLI or the Autonomous Database REST APIs. This option is not available using the Oracle Cloud Infrastructure Console.

To create a cross tenancy clone:

1. Perform the prerequisite steps to define the OCI Identity and Access Management policies to authorize cross tenancy cloning.

   See Prerequisites for Cross Tenancy Cloning for details.

2. On the tenancy where you want to create the clone, on the destination tenancy in the destination region, use the CLI or call the REST API with a valid clone type FULL or METADATA and provide the OCID of the source database, where the source database resides in a different tenancy (the source tenancy).

   For example, with the CLI:

   ```
   oci db autonomous-database create-from-clone
         --clone-type metadata
         --compartment-id ocid1.tenancy.oc1..unique_ID
         --source-id ocid1.autonomousdatabase.oc1.iad.unique_ID
         --db-name dbnameclone
         --admin-password password
         --data-storage-size-in-tbs 1
         --compute-model ECPU
         --compute-count 4
   ```

   See create-from-clone for more information.

   Use the `CreateAutonomousDatabase` API to create a cross tenancy clone.

   See the following for additional information on the REST API:

   - CreateAutonomousDatabase

- CreateAutonomousDatabaseCloneDetails

- For information about using the API and signing requests, see REST APIs and Security Credentials.
- For information about SDKs, see Software Development Kits and Command Line Interface.

Create a Cross Tenancy Clone from a Backup

Shows the steps to create a cross tenancy clone from a backup.

These steps cover creating a Full clone or a Metadata clone. See Create a Cross Tenancy or Cross-Region Refreshable Clone for details on creating a cross tenancy refreshable clone.

> **Note:**
>
> The cross tenancy cloning option is only available using the CLI or the Autonomous Database REST APIs. This option is not available using the Oracle Cloud Infrastructure Console.

To create a cross tenancy clone from a backup:

1. Perform the prerequisite steps to define the OCI Identity and Access Management policies to authorize cross tenancy cloning.

   See Prerequisites for Cross Tenancy Cloning for details.

2. On the tenancy where you want to create the clone, on the destination tenancy in the destination region, use the CLI or call the REST API with a valid clone type FULL or METADATA and provide the OCID of the backup (on the source tenancy), where the source database resides in a different tenancy (the source tenancy).

   > **Note:**
   >
   > See Create a Cross Tenancy or Cross-Region Refreshable Clone to create a cross tenancy refreshable clone.

   For example, with the CLI:

   ```
   oci db autonomous-database create-from-backup-timestamp
        --autonomous-database-id ocid1.autonomousdatabase.oc1.iad.anuw_example
        --clone-type full
        --compartment-id ocid1.tenancy.oc1..fcue4_example
        --admin-password password
        --compute-model ECPU
        --compute-count 2
        --db-name ExampleTest1
        --timestamp 2023-12-15T19:30:00Z
        --data-storage-size-in-tbs 1
   ```

   See create-from-backup-timestamp and create-from-backup-id for more information.

Use the `CreateAutonomousDatabase` API to create a cross tenancy clone by cloning from a backup of an existing Autonomous Database.

See the following for information on the REST API:

- CreateAutonomousDatabase
- CreateAutonomousDatabaseFromBackupDetails Reference
- CreateAutonomousDatabaseFromBackupTimestampDetails Reference

- For information about using the API and signing requests, see REST APIs and Security Credentials.
- For information about SDKs, see Software Development Kits and Command Line Interface.

Clone Autonomous Database to Change Workload Type

When you create an Autonomous Database clone, by default the clone is of the same workload type. Optionally, you can select that the clone is of a different workload type than the source database.

Cloning a database to a different workload type is useful when:

- You have provisioned a database of an incorrect workload type and now need to clone the database to the appropriate workload type.
- You need a copy of your database of a different workload type to branch out to run new types of workloads on your data.

Note the following when you clone to a different workload type:

- You can clone between the following workload types:
  - Autonomous Data Warehouse to Autonomous Transaction Processing
  - Autonomous Transaction Processing to Autonomous Data Warehouse
  - Autonomous JSON Database to Autonomous Transaction Processing or Autonomous Data Warehouse or APEX Service
  - APEX Service to Autonomous JSON Database or Autonomous Transaction Processing or Autonomous Data Warehouse
- You may also clone from a backup, when choosing to clone a database to a different workload type.
- The cloned database can be in the same region or in a different region (cross-region).
- Cloning a database to a different workload type does not affect the database's existing data dictionary, optimizer statistics, and object compression. New objects that are created after cloning are created using the cloned database's workload type parameters (for example compression and statistics).
- When you clone from Autonomous Data Warehouse to Autonomous Transaction Processing, the cloned database has the additional services TP and TPURGENT.
- When you clone from one of the following to Autonomous Data Warehouse the cloned database does not have the services TP or TPURGENT:
  - Autonomous Transaction Processing
  - Autonomous JSON Database
  - APEX Service

The following are not allowed when you clone and select a different workload type:

- Creating a cross workload refreshable clone.
- Cloning to a more restrictive workload type, such as from Autonomous Transaction Processing or Autonomous Data Warehouse database to Autonomous JSON Database or APEX Service.

See the following for steps to create an Autonomous Database clone:

- Clone an Autonomous Database Instance
- Clone an Autonomous Database from a Backup

Notes for Cloning Autonomous Database

Provides information about the cloning operation and the resulting cloned database.

- General Notes for Cloning on Autonomous Database
  Provides general information about the cloning operation and the resulting cloned database.
- Notes for Cross Tenancy and Cross Region Cloning
  Provides information about cross tenancy and cross region cloning.
- Resource Management Rules and Performance Data for a Cloned Database
  Provides notes for resource management rules and performance data for a cloned database.
- Optimizer Statistics for a Cloned Database
  During the provisioning for either a Full Clone or a Metadata Clone, the optimizer statistics are copied from the source database to the cloned database.
- Disable Oracle Scheduler Jobs for a Cloned Database
  Oracle Scheduler jobs from a source database are copied to the clone when you perform a clone operation. You can assure that cloned Oracle Scheduler jobs are disabled on the clone.

General Notes for Cloning on Autonomous Database

Provides general information about the cloning operation and the resulting cloned database.

- If there is an ongoing scaling operation on a source database, you cannot initiate a clone operation until the ongoing operation completes. If you attempt such an operation you will see message such as the following:

  ```
  The operation cannot be performed because the Autonomous Database with Id
  **** is in the SCALE_IN_PROGRESS state.
  ```

- If you define a network Access Control List (ACL) on the source database, the currently set network ACL is cloned to the new database. If a database is cloned from a backup, the current source database's ACL is applied (not the ACL that was valid at the time of the backup).
- If you create a clone and the source database has an access control list (ACL) and you specify the private endpoint network access option, **Virtual cloud network** for the target database, the ACL is not cloned to the new database. In this case, you must define security rules within your Network Security Group (or groups) to control traffic to and from your target database (instead of using the access control rules that were specified in the ACL on the clone source). See Configure Private Endpoints When You Provision or Clone an Instance for more information.

- Cloning an Autonomous Database instance copies your database files to a new instance. There is no downtime associated with cloning and the cloning operation has no impact on applications running on the source.
- For a **Metadata Clone**, the APEX Apps and the OML Projects and Notebooks are copied to the clone. For a **Metadata Clone**, the underlying database data of the APEX App or OML Notebook is not cloned.
- The Autonomous Database Details page for an Autonomous Database instance that was created by cloning includes the **Cloned From** field. This displays the name of the database where the clone was created.

Notes for Cross Tenancy and Cross Region Cloning

Provides information about cross tenancy and cross region cloning.

- For cloning, when you choose a region other than the current region for your clone target, using either the Oracle Cloud Infrastructure CLI or Terraform you can only perform such a cross-region clone from the remote region. That is, call the create clone API from the remote region to which you want to clone, with the source database OCID as that of the source that you want to clone from.
- The cross tenancy cloning option is only available using the CLI or the Autonomous Database REST APIs. Cross tenancy cloning is not available using the Oracle Cloud Infrastructure Console.

  See Cross Tenancy and Cross-Region Cloning for more information.
- The cross tenancy cloning option does not support cloning with customer managed keys on the source. See Manage Encryption Keys on Autonomous Database for more information on customer managed keys.

Resource Management Rules and Performance Data for a Cloned Database

Provides notes for resource management rules and performance data for a cloned database.

The following applies for resource management rules and performance data in a cloned database:

- During the provisioning for either a Full Clone or a Metadata Clone, any resource management rule changed by the user in the source database is carried over to the cloned database.
- For a cloned database, performance data for the time prior to the clone operation is not visible in the **Database Dashboard** card (under Monitor in Database Actions).

For more information on setting resource management rules, see Manage Runaway SQL Statements on Autonomous Database.

Optimizer Statistics for a Cloned Database

During the provisioning for either a Full Clone or a Metadata Clone, the optimizer statistics are copied from the source database to the cloned database.

The following applies for optimizer statistics for tables in a cloned database:

- Full Clone: loads into tables behave the same as loading into a table with statistics already in place.
- Metadata Clone: the first load into a table after the database is cloned clears the statistics for that table and updates the statistics with the new load.

For more information on Optimizer Statistics, see Optimizer Statistics Concepts.

Disable Oracle Scheduler Jobs for a Cloned Database

Oracle Scheduler jobs from a source database are copied to the clone when you perform a clone operation. You can assure that cloned Oracle Scheduler jobs are disabled on the clone.

To disable Oracle Scheduler jobs on Autonomous Database instances that are cloned, you can define a trigger using the `ON CLONE` clause. To use the `ON CLONE` clause, define the trigger on the source database before you perform the clone operation.

For example:

```
CREATE OR REPLACE TRIGGER after_clone_instance
     AFTER CLONE
     ON PLUGGABLE DATABASE
BEGIN
-- Disable specific jobs or use a cursor to grab all scheduled jobs and
disable them
     DBMS_SCHEDULER.DISABLE(name=> job_name);
END;
/
```

Move an Autonomous Database to a Different Compartment

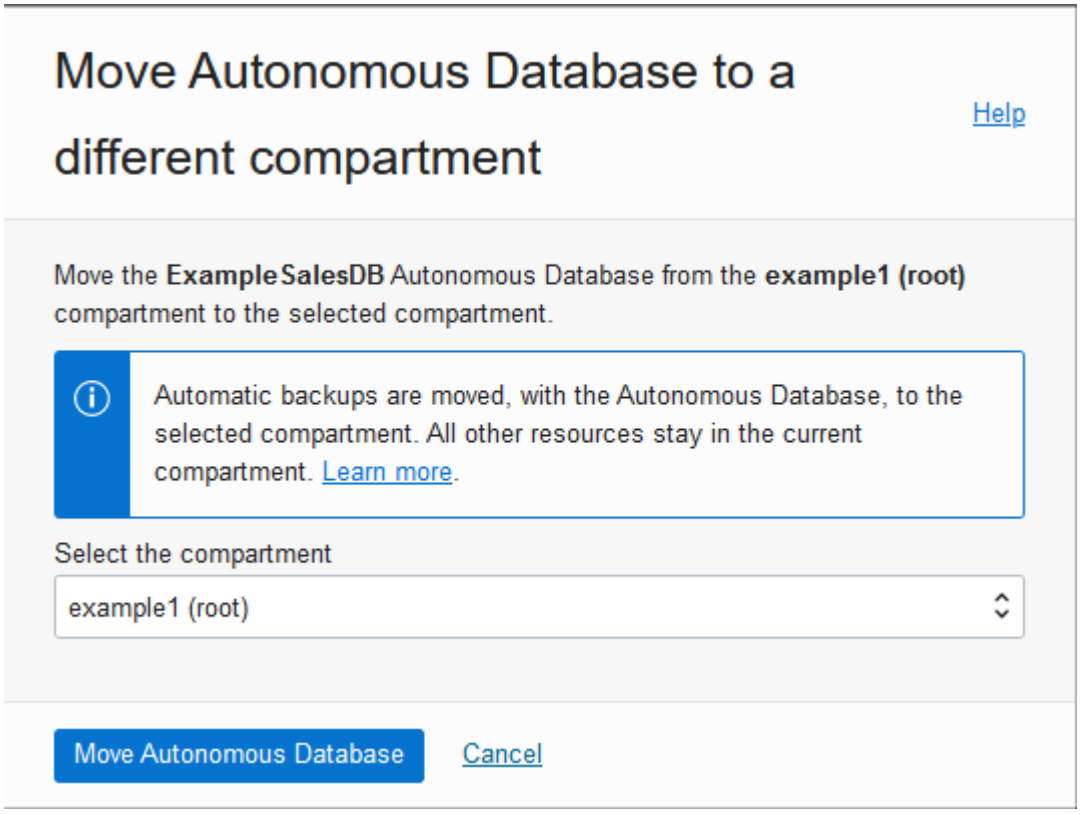
Shows you the steps to move a database to a different Oracle Cloud Infrastructure compartment.

- To move a database you must have the right to manage Autonomous Databases in the database's current compartment and in the compartment you are moving it to.
- As soon as you move a database to a different compartment, the policies that govern the new compartment apply immediately and affect access to the database. Therefore, your access to the database may change, depending on the policies governing your Oracle Cloud user account's access to resources.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.

1. Choose your region. See Switching Regions for information on switching regions and working in multiple regions.
2. Choose your **Compartment**. See Compartments for information on using and managing compartments.
3. Select an Autonomous Database instance from the list in your compartment.
4. On the **Details** page, from the **More actions** drop-down list, select **Move resource**.
5. In the **Move Autonomous Database to a different compartment** page, select the new compartment.



6. Click **Move Autonomous Database**.

See Moving Database Resources to a Different Compartment for more information.

Use Refreshable Clones with Autonomous Database

Autonomous Database provides cloning where you can choose to create a full clone of the active instance, create a metadata clone, or create a refreshable clone. With a refreshable clone the system creates a clone that can be easily updated with changes from the source database.

For details on creating a clone that is a full clone or a metadata clone, see Cloning and Moving an Autonomous Database.

- About Refreshable Clones on Autonomous Database
- Prerequisites for Creating a Refreshable Clone
  Describes prerequisites for creating a refreshable clone.
- Create a Refreshable Clone for an Autonomous Database Instance
  Shows you the steps to create an Autonomous Database refreshable clone from the Oracle Cloud Infrastructure Console.
- View Refreshable Clones for an Autonomous Database Instance
- Edit Automatic Refresh Policy for Refreshable Clone
  When you enable the automatic refresh option a refreshable clone automatically refreshes from the source database at regular intervals. By default, automatic refresh is disabled and you must manually refresh at least once every 7 days.

- Refresh a Refreshable Clone on Autonomous Database
  Shows you the steps to refresh a refreshable clone from the Oracle Cloud Infrastructure Console.
- Disconnect a Refreshable Clone from the Source Database
- Reconnect a Refreshable Clone to the Source Database
- Create a Cross Tenancy or Cross-Region Refreshable Clone
- Use the API
  Provides details for using the APIs to create, refresh, list, disconnect, and reconnect a refreshable clone.
- Refreshable Clone Notes
  Lists limitations and notes for Autonomous Database refreshable clones.

About Refreshable Clones on Autonomous Database

When you create a refreshable clone for an Autonomous Database instance the system clones the source database to a database that can be refreshed from the source.
Refreshable clones are billed based on their base CPU count, plus any additional CPU usage if compute auto scaling is enabled; they do not get billed additionally for the CPUs of the source database. The number of base CPUs is specified by the number of ECPUs (OCPUs if your database uses OCPUs) as shown in the **ECPU count** or **OCPU count** field on the Oracle Cloud Infrastructure Console. See Oracle Autonomous Database Serverless Features Billing for more information.

Using Cloud Links you can specify that access to a data set from one or more databases is offloaded to a refreshable clone. When a consumer Autonomous Database is listed in a data set's offload list, access to the data set is directed to the refreshable clone. See Register a Data Set with Offload Targets for Data Set Access for more information.

- Refreshable Clone Features
  Describes refreshable clone features.
- Refreshable Clone Operations
  You can create a refreshable clone from an Autonomous Database instance. After you create a refreshable clone you can perform several operations on the refreshable clone, including: refresh, stop, start, restart, disconnect from source, and terminate.
- Refreshable Clone with Automatic Refresh Enabled
  By default you manually refresh a refreshable clone with changes from the source database. When you enable the automatic refresh option, Autonomous Database automatically refreshes a refreshable clone with data from the source database at given time intervals.
- Refreshable Clone Lifecycle States
  After you create a refreshable clone, the clone indicates its state on the Autonomous Database Information page in the Lifecycle State field. In addition, the Mode field indicates that a refreshable clone is **Read-Only**.
- Refreshable Clone Refresh Timing and Disconnecting from the Source Database
  A refreshable clone with **Automatic refresh** disabled has a one week refresh age limit and a banner on the Oracle Cloud Infrastructure console displays the date and time up to which you can refresh the refreshable clone. The banner also includes a **Refresh clone** button.
- Refreshable Clone Reconnect to the Source Database
  After you perform a **Disconnect refreshable clone** operation, a banner displays the date and time up to which you can reconnect the database to the source. The banner also includes a **Reconnect refreshable clone** button.

- Operations on an Autonomous Database with an Attached Refreshable Clone
  Describes details for using a source Autonomous Database instance that has one or more attached refreshable clones.

Refreshable Clone Features

Describes refreshable clone features.

A refreshable clone allows you to do the following:

- Maintain one or more copies of the source database for use as read-only databases. A clone database is available when you need it, and when you want to update the data you can refresh the clone from the source database.
- Create one or more clones in regions other than the region of your primary (source) database. Clones in remote regions can be refreshed from the source database.
- You can enable the refreshable clone automatic refresh option. With automatic refresh enabled Autonomous Database automatically refreshes the clone with data from the source. See Refreshable Clone with Automatic Refresh Enabled for more information.
- Share copies of a production database with multiple business units. For example, one business unit might use the source database for ongoing transactions and another business unit could at the same time use the refreshable clone database for read-only operations.

  This option also allows you to spread the cost of database usage across multiple business units. You can bill the different units separately, based on their usage of one or more refreshable clone databases.
- Use a refreshable clone as a test database. You can disconnect a refreshable clone from its source and perform DML operations or calculations as needed, in addition to querying data. This allows you to run DML and make changes while the database is disconnected. When you are done with your testing you can reconnect to the source database, which refreshes the clone to the point where it was when you disconnected.

  The reconnect operation is only available for 24 hours after the disconnect time. If you do not reconnect within the reconnect period, the clone is disconnected from the source database and refreshing and reconnecting are not possible.

> **Note:**
>
> Refreshable clones have a one week refresh age limit. If you do not perform a refresh within a week, the refreshable clone is no longer refreshable. After a refreshable clone passes the refresh time limit, you can use the instance as a read only database or you can disconnect from the source to make the database a read/write (standard) database.

Refreshable Clone Operations

You can create a refreshable clone from an Autonomous Database instance. After you create a refreshable clone you can perform several operations on the refreshable clone, including: refresh, stop, start, restart, disconnect from source, and terminate.

Operation	Description
Create	You can create a refreshable clone from an Autonomous Database instance. You can create more than one refreshable clone using the same Autonomous Database instance as a source. See Create a Refreshable Clone for an Autonomous Database Instance for the steps to create a refreshable clone.
View	You view a refreshable clone from the Oracle Cloud Infrastructure console Autonomous Database Details page. See View Refreshable Clones for an Autonomous Database Instance for more information.
Start or Restart	When a refreshable clone is stopped as indicated by the Lifecycle State **Stopped**, you can start the database. When a refreshable clone is available as indicated by the Lifecycle State **Available**, you can restart the database.
Refresh	For a refreshable clone, you can refresh the clone with data from the source database. See Refresh a Refreshable Clone on Autonomous Database for more information.
Edit Automatic Refresh Policy	When you enable the automatic refresh option a refreshable clone automatically refreshes from the source database at regular intervals. By default, automatic refresh is disabled and you must manually refresh at least once every 7 days. See Edit Automatic Refresh Policy for Refreshable Clone for more information.
Disconnect Clone from Source	You can disconnect a refreshable clone from the source database to make the clone a standard read/write database. See Disconnect a Refreshable Clone from the Source Database for more information.
Reconnect Refreshable Clone	When a clone database is disconnected, you can use the clone as a standard read/write database. There is a 24 hour reconnect period where you can reconnect the clone to its source database. After the reconnect period the refreshable clone is disassociated from the source database and reconnecting to the source database is not allowed. See Reconnect a Refreshable Clone to the Source Database for more information.
Stop	When a refreshable clone is stopped, database operations are not available and charging for OCPU usage on the refreshable clone stops.
Terminate	If you want to terminate a refreshable clone, select **More actions** and **Terminate**. Terminating a refreshable clone disassociates the clone database from the source database.

Refreshable Clone with Automatic Refresh Enabled

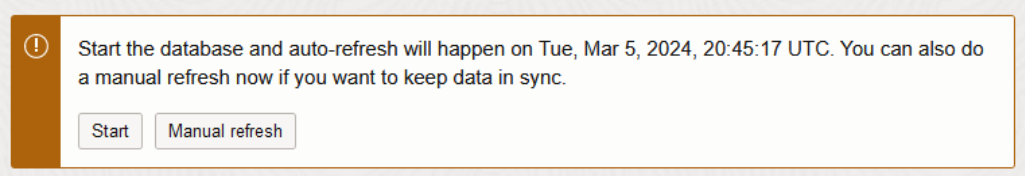
By default you manually refresh a refreshable clone with changes from the source database. When you enable the automatic refresh option, Autonomous Database automatically refreshes a refreshable clone with data from the source database at given time intervals.

> **Note:**
>
> The automatic refresh option is only available when the source database uses the ECPU compute model.

Note the following for a refreshable clone with automatic refresh enabled:

- You can manually refresh a refreshable clone between automatic refreshes. If you manually refresh to a refresh point later than the next refresh point specified for an automatic refresh, the automatic refresh may fail. Following refreshes, after the failed refresh, are attempted at the next scheduled automatic refresh.
- If a refreshable clone is stopped, disconnected, or already at a later timestamp than the specified automatic refresh point, the automatic refresh fails.
- If a scheduled refresh is missed due to the source database being stopped, or the refreshable clone being disconnected, or when the refreshable clone was manually refreshed to later point, the missed refresh is skipped. The refreshable clone refreshes at the next scheduled interval.
- When an automatic refresh is missed, Autonomous Database does the following:
  - Shows a banner on the Oracle Cloud Infrastructure Console.

    For example:

  - Generates the automatic refresh failed event. See Autonomous Database Event Types for more information.
- If no refresh is performed within 7 days, the refreshable clone can no longer be refreshed.

  When a refreshable clone has not been refreshed within seven days and the refreshable clone has exceeded its maximum refresh time, you have the following options:
  - You can continue to use the refreshable clone as a read-only database. The refreshable clone is not refreshable and the data on the refreshable clone reflects the state of the source database at the time of the last successful refresh.
  - You can disconnect the refreshable clone from the source database. This disconnects the refreshable clone from the source Autonomous Database instance.

    See Disconnect a Refreshable Clone from the Source Database for more information.

Refreshable Clone Lifecycle States

After you create a refreshable clone, the clone indicates its state on the Autonomous Database Information page in the Lifecycle State field. In addition, the Mode field indicates that a refreshable clone is **Read-Only**.

A refreshable clone indicates its state as follows:

- **Updating**: When a refreshable clone is refreshing or reconnecting, the Lifecycle State field shows **Updating**. While the database is refreshing connections and queries wait until the refresh completes. After the refresh completes the state is set to **Available**, and connections and queries resume.

  See Refresh a Refreshable Clone on Autonomous Database for more information.
- **Stopped**: When a refreshable clone is stopped, database operations are not available and charging for OCPU usage on the refreshable clone stops.
- **Available**: When the refreshable clone is available, database operations are available and you are charged for OCPU usage on the refreshable clone.

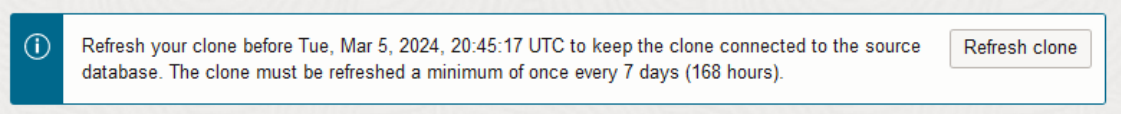
The Autonomous Database Information page Mode field indicates the database mode, as follows:

- **Read-Only**: No data can be inserted into or updated in a refreshable clone as it is a read-only database. You can use a refreshable clone for read-only queries and for reporting.

  See Disconnect a Refreshable Clone from the Source Database to change the database to Read/Write mode. In this case the refreshable clone is disconnected from the source database.

Refreshable Clone Refresh Timing and Disconnecting from the Source Database

A refreshable clone with **Automatic refresh** disabled has a one week refresh age limit and a banner on the Oracle Cloud Infrastructure console displays the date and time up to which you can refresh the refreshable clone. The banner also includes a **Refresh clone** button.



When a refreshable clone that has **Auto refresh** disabled is not refreshed within seven (7) days from the last refresh, the banner message changes to indicate that the refreshable clone that has not been refreshed within seven days cannot be refreshed. The button in the banner changes to **Disconnect clone from source database**.

When a refreshable clone has not been refreshed within seven days and the refreshable clone has exceeded its maximum refresh time, you have the following options:

- You can continue to use the refreshable clone as a read-only database. The refreshable clone is not refreshable and the data on the refreshable clone reflects the state of the source database at the time of the last successful refresh.
- You can disconnect the refreshable clone from the source database. This disconnects the refreshable clone from the source Autonomous Database instance.

  See Disconnect a Refreshable Clone from the Source Database for more information.

When a refreshable clone has exceeded the maximum refresh time, if you want to use a refreshable clone that can be refreshed from the source database, you must create a new refreshable clone. If you create a new refreshable clone, you might also want to terminate the refreshable clone that is no longer able to refresh from the source database.

Refreshable Clone Reconnect to the Source Database

After you perform a **Disconnect refreshable clone** operation, a banner displays the date and time up to which you can reconnect the database to the source. The banner also includes a **Reconnect refreshable clone** button.

When a disconnected database is not reconnected within 24 hours from the disconnect time, the Oracle Cloud Infrastructure Console removes the reconnect refreshable clone banner.

When a disconnected refreshable clone has exceeded the reconnect period, you have the following options:

- You can use the database as a standard Autonomous Database; there is no longer an option to reconnect the database to the source database.

- If you want to use a refreshable clone that can be refreshed from the source database, then you must create a new refreshable clone. If you create a new refreshable clone, then you might want to terminate the disconnected clone that is no longer able to refresh from the source database.

Operations on an Autonomous Database with an Attached Refreshable Clone

Describes details for using a source Autonomous Database instance that has one or more attached refreshable clones.

When you make certain changes on a source Autonomous Database instance that has one or more refreshable clones attached to it, the changes are applied to both the source database and to the refreshable clones as follows:

- Storage: The storage value you set on the source database applies to both the source database and to any attached refreshable clones.
- ADMIN password: The ADMIN password value you set on the source database applies to both the source database and to any attached refreshable clones.

To view the refreshable clones for a source database, on the Autonomous Database Details page, under **Resources**, click **Refreshable Clones**. The Autonomous Database resources area provides a link to each refreshable clone in the Display Name field, and includes the Last Refresh timestamp field and the Refresh Point timestamp field. The refresh point specifies the timestamp for the source database data to which the refreshable clone data is refreshed.

If you want to terminate a source database that has one or more attached refreshable clones, then before you terminate the source database you must do the following until there are no longer any attached refreshable clones. For each attached refreshable clone, do one of the following:

- Disconnect the refreshable clone from the source database. See Disconnect a Refreshable Clone from the Source Database for more information.
- Terminate the refreshable clone to disassociate the refreshable clone from the source database. You can terminate a refreshable clone by selecting **More actions** and **Terminate**.

Prerequisites for Creating a Refreshable Clone

Describes prerequisites for creating a refreshable clone.

To create a refreshable clone you must define the required access using OCI Identity and Access Management policy statements written by an administrator, whether you're using the Console, the REST API, the CLI, or another tool.

The following policies allow you to create a refreshable clone:

```
Allow group Group_Name to read autonomous-databases in compartment
Compartment_Name
      where target.id = 'oc1.autonomousdatabase.oc1..unique_ID'


Allow group Group_Name to manage autonomous-databases in compartment
Compartment_Name
```

The `where` clause is optional and provides a more fine grained way to grant access to a specific database.

You can limit cloning permissions so that the group can only clone Autonomous Databases but cannot create Autonomous Databases, or further limit permission to only create a particular type of clone: Full Clone, Metadata Clone, or Refreshable Clone. See IAM Permissions and API Operations for Autonomous Database for more information and examples.

This shows the policies for cloning within a tenancy. See Prerequisites for Cross Tenancy Cloning for cross tenancy cloning policies.

See IAM Policies for Autonomous Database and Getting Started with Policies for more information.

Create a Refreshable Clone for an Autonomous Database Instance

Shows you the steps to create an Autonomous Database refreshable clone from the Oracle Cloud Infrastructure Console.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To create a refreshable clone do the following:

1. On the **Details** page, from the **More actions** drop-down list, select **Create clone**.
2. On the **Create Autonomous Database clone** page, choose the clone type **Refreshable clone** from the choices:
   - **Full clone**: creates a new database with the source database's data and metadata.
   - **Refreshable clone** To create a refreshable clone, select this clone type.
   - **Metadata clone**: creates a new database with the source database's metadata without the data.
3. Provide basic information for the Autonomous Database clone.
   - **Choose your preferred region**. Use the current region or select a different region. When you create a refreshable clone, the preferred region list only shows a remote region if your tenancy is subscribed to the remote region (you must be subscribed to the remote region where you create a refreshable clone).

     The list of available regions only shows a remote region if your tenancy is subscribed to the remote region (you must be subscribed to the paired remote region for the target region where you are cloning your database). See Autonomous Database Cross-Region Paired Regions for more information.
   - **Create in compartment**. See Compartments for information on using and managing compartments.
   - **Source database name** This field is read-only and shows the name of the source database.
   - **Display name** Specify a user-friendly description or other information that helps you easily identify the database.

You can use the name provided, of the form: **Clone-of-***DBname* or change this to the name you want to use to identify the database. The supplied *DBname* is the name of the source database that you are cloning.

- **Database name**: Specify the database name; it must consist of letters and numbers only. The maximum length is 30 characters. The same database name cannot be used for multiple Autonomous Databases in the same tenancy in the same region.

  The default database name is a generated 16-character string.

4. Configure the database (with ECPU compute model)

   - **ECPU count**: Specify the number of ECPUs for your database. The minimum value for the number of ECPUs is 2.

     Your license type determines the **ECPU count** maximum. For example, if your license type is Bring your own license (BYOL) with Oracle Database Standard Edition (SE), the **ECPU count** maximum is 32.

   - **Compute auto scaling**: By default compute auto scaling is enabled to allow the system to automatically use up to three times more CPU and IO resources to meet workload demand. If you do not want to use compute auto scaling then deselect this option.

     See Use Auto Scaling for more information.

   - **Show advanced options**: Click to show the compute model options or if you want to create or join an elastic pool:

     – **Enable elastic pool**:

       See Create or Join an Elastic Pool While Provisioning or Cloning an Instance for more information.

     – **Compute model**: Shows the selected compute model.

> **Note:**
>
> The storage for a refreshable clone is set to the same size as on the source database. To change the storage size for a refreshable clone, you must change the storage value on the source database.

5. Select **Enable automatic refresh** to specify that the refreshable clone automatically refreshes. By default, **Enable automatic refresh** is deselected and you must manually refresh at least once every 7 days. When you select **Enable automatic refresh**, the dialog shows two fields:

   - **Frequency of refresh**: specifies the refresh frequency in hours or days. The minimum is one hour and the maximum is 7 days. The default **Frequency of refresh** value is 1 hour.

   - **Data lag**: specifies the data lag in minutes, hours, or days, the minimum is 0 minutes and the maximum is 7 days. This is the value that specifies the amount of time behind the source that the data refresh is, where a value of 0 specifies that the refreshable clone refreshes to the latest available timestamp. The default **Data lag** value is 0.

6. Choose network access

> **Note:**
>
> After you clone your Autonomous Database you can change the network access option you select for the cloned instance.

- **Secure access from everywhere**

  By default all secure connections are allowed from everywhere.

- **Secure access from allowed IPs and VCNs only**

  This option restricts connections to the database according to the access control rules (ACLs) you specify. To add multiple ACLs for the Autonomous Database, select this option and click **Add access control rule**.
  See Configure Access Control Lists When You Provision or Clone an Instance for more information.

- **Private endpoint access only**

  This option assigns a private endpoint, private IP, and hostname to your database. Specifying this option allows traffic only from the VCN you specify; access to the database from all public IPs or VCNs is blocked. This allows you to define security rules, ingress/egress, at the Network Security Group (NSG) level and to control traffic to your Autonomous Database.

  See Configure Private Endpoints When You Provision or Clone an Instance for more information.

7. Choose license and Oracle Database edition

   - **License included**

     The default is the license included license type. With this option you subscribe to new database software licenses and the database cloud service.

   - **Switch to Bring your own license (BYOL)**

     Select this to show more license options. Select this if your organization already owns Oracle Database software licenses and you want to bring your existing database software licenses to the database cloud service. See Cloud pricing for information on Bring your own license (BYOL) and other licensing options for Oracle Cloud Infrastructure cloud service pricing.

     When you select to switch to **Bring your own license (BYOL)**, you also select an Oracle Database edition. The Oracle Database edition you select is based on the license you bring to Autonomous Database and changes the maximum value that you can select for the **ECPU count**. The choices are:

     – **Oracle Database Enterprise Edition (EE)**: For this license type the maximum allowed value for **ECPU count** is 512, however you may contact your Oracle account team to request more ECPUs. With compute auto scaling enabled you can use up to **ECPU count** x 3 ECPUs. For example, if you set the **ECPU count** to 512, you can use up to 1,536 ECPUs.

     – **Oracle Database Standard Edition (SE)**: For this license type the maximum allowed value for **ECPU count** is 32. With compute auto scaling enabled you can use up to **ECPU count** x 3 ECPUs. This license restricts the number of ECPUs you can use to a maximum of 32 ECPUs, with or without compute auto scaling enabled.

       See Use Auto Scaling for more information.

8. (Optional) Provide contacts for operational notifications and announcements

   Click **Add contact** and in the **Contact email** field, enter a valid email address. If the database you are cloning has a customer contact list, the list is copied. To enter multiple **Contact email** addresses, repeat the process to add up to 10 customer contact emails.

   See View and Manage Customer Contacts for Operational Issues and Announcements for more information.

9. (Optional) Click Show advanced options to select advanced options.

   - **Encryption key**
     A refreshable clone uses the encryption key of the source Autonomous Database.

     See Use Customer-Managed Encryption Keys on Autonomous Database for more information.

   - **Maintenance**
     Patch level By default the patch level is the patch level of the source database. Select **Early** to configure the instance with the early patch level. When cloning a source database with **Early** patch level, you can only choose the **Early** patch level for your clone.

     See Set the Patch Level for more information.

   - **Management**
     Shows the character set and national character set for your database.

     See Choose a Character Set for Autonomous Database for more information.

   - **Tools**
     Refreshable clones inherit their tool status for each built-in tool from the refreshable clone's source database. If the source of a refreshable clone changes its tool configuration, the change is reflected in the refreshable clone after the next refresh.

     See Configure Autonomous Database Built-in Tools for more information.

   - **Tags**
     If you want to use Tags, enter the **Tag key** and **Value**. Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values which can be attached to resources.

     See Tagging Overview for more information.

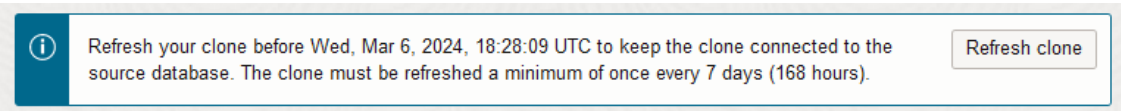
10. Click **Create Autonomous Database clone**.

On the Oracle Cloud Infrastructure console the **State** shows **Provisioning...** until the refreshable clone is available.

After the provisioning completes the Lifecycle state shows **Available** and the Mode is **Read-only**.

After you download the wallet for the refreshable clone and connect to the database, you can begin using the database to perform read-only operations such as running queries or building reports and notebooks.

When **automatic refresh** is disabled, after you create a refreshable clone, the Oracle Cloud Infrastructure console shows a banner similar to the following with a message indicating the date prior to which the next refresh must be completed (the banner shows the 7 day refresh limit).



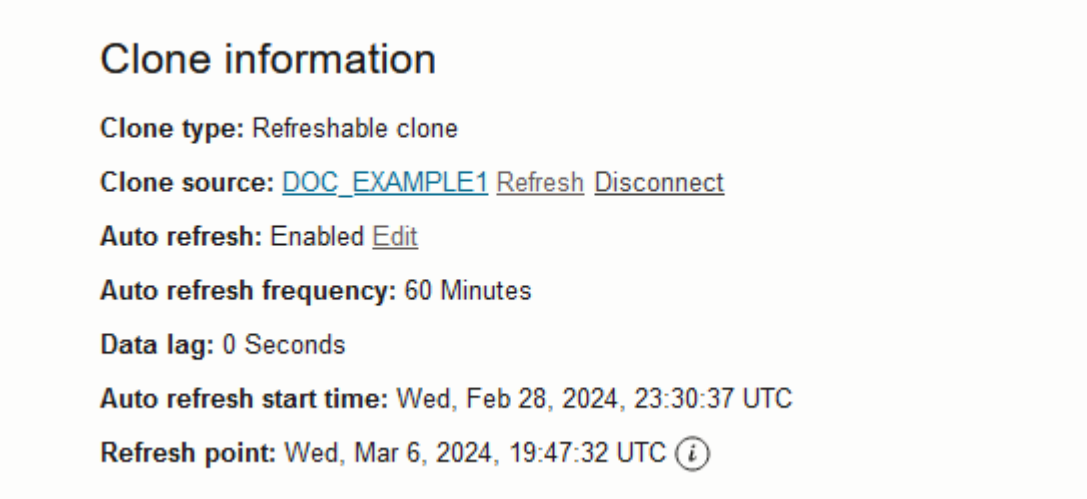
See Refresh a Refreshable Clone on Autonomous Database for details on refreshing a refreshable clone.

View Refreshable Clones for an Autonomous Database Instance

If you know a refreshable clone's display name, you can view the instance by selecting from the list of Autonomous Databases on the Oracle Cloud Infrastructure Console.

On the Oracle Cloud Infrastructure Console, the details page for a refreshable clone shows a **Refreshable clone** tag to indicate the instance is a refreshable clone.

The **Clone information** area shows details for a refreshable clone. You can access the source database console by clicking the link in the **Clone source** field.



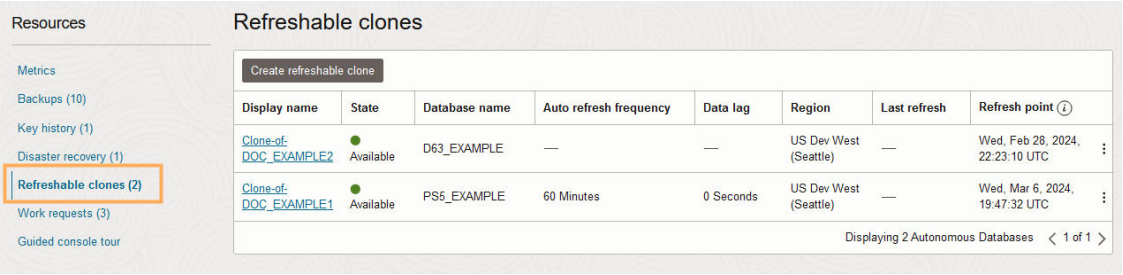
On the details under **Clone information** the console shows the **Refresh point**, indicating the timestamp of the source database's data that the clone was last refreshed with.

If an Autonomous Database has attached refreshable clones, you can view the refreshable clones as follows:

1. Choose your region. See Switching Regions for information on switching regions and working in multiple regions.
2. Choose your **Compartment**. See Compartments for information on using and managing compartments.
3. Select an Autonomous Database instance from the list in your compartment.
4. On the **Autonomous Database details** page, under **Resources**, click **Refreshable clones**.

This shows the list of refreshable clones for the Autonomous Database instance.

If you click  at the end of a row for a refreshable clone in the list, you can select an action to perform for the refreshable clone.



Edit Automatic Refresh Policy for Refreshable Clone

When you enable the automatic refresh option a refreshable clone automatically refreshes from the source database at regular intervals. By default, automatic refresh is disabled and you must manually refresh at least once every 7 days.

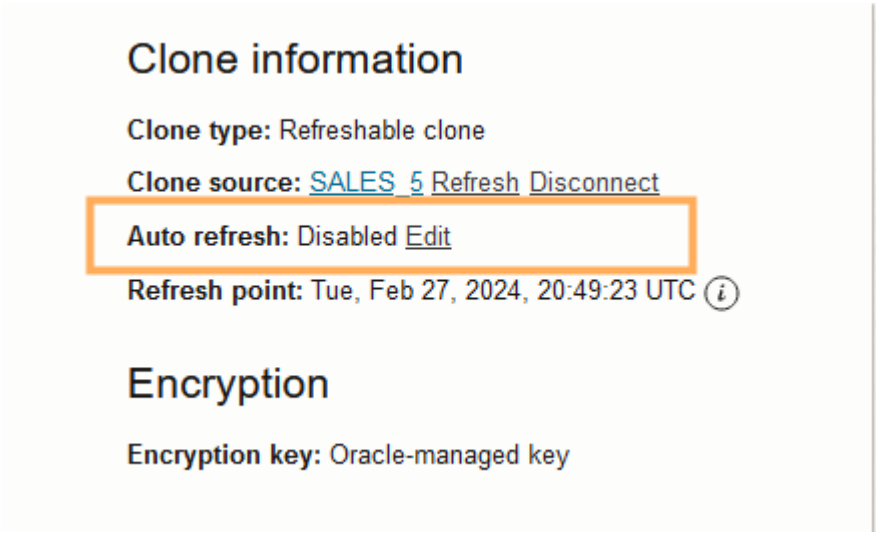
The edit auto refresh settings dialog allows you to enable or disable automatic refresh for a refreshable clone. When automatic refresh is enabled you can change the automatic refresh options.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To edit auto refresh options:

1. On the clone source, on the **Details** page, under **Resources**, select **Refreshable clones**.
2. In the Refreshable clones area, under **Display name**, select a refreshable clone.
3. On the refreshable clone, under the **Clone information** area, in the **Auto refresh** field click **Edit**.



This shows the Edit auto refresh settings page.

4. Select or deselect **Enable automatic refresh** to enable or disable automatic refresh for the refreshable clone. When **Enable automatic refresh** is selected, you can modify the options:
   - **Frequency of refresh**: specifies the refresh frequency in hours or days. The minimum is one hour and the maximum is 7 days. The default **Frequency of refresh** value is 1 hour.
   - **Data lag**: specifies the data lag in minutes, hours, or days, the minimum is 0 minutes and the maximum is 7 days. This is the value that specifies the amount of time behind the source that the data refresh is, where a value of 0 specifies that the refreshable clone refreshes to the latest available timestamp. The default **Data lag** value is 0.
5. Click **Save**.

Refresh a Refreshable Clone on Autonomous Database

Shows you the steps to refresh a refreshable clone from the Oracle Cloud Infrastructure Console.

Perform the following prerequisite steps as necessary:

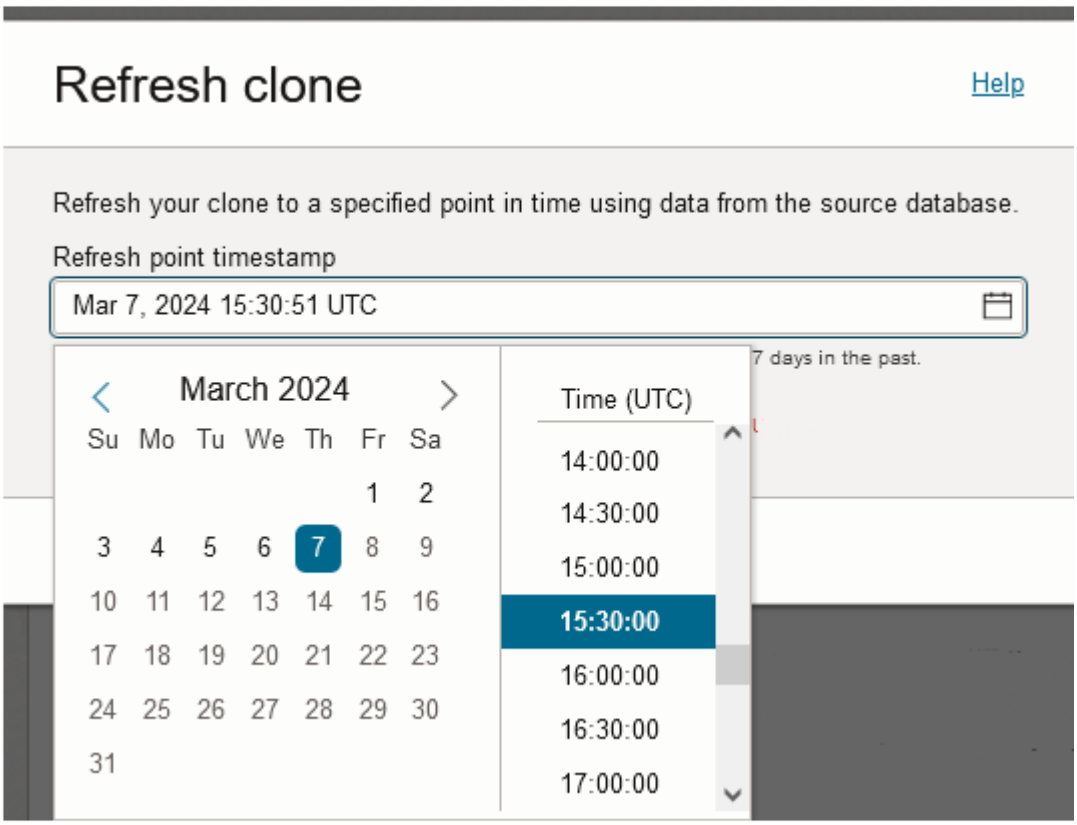
- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

You can refresh a refreshable clone as follows:

1. On the Details page, under Clone information, in the **Clone source** field click the **Refresh** link.

   This shows the Refresh clone dialog.
2. In the Refresh clone dialog, specify the **Refresh point timestamp**.

   The refresh point specifies the timestamp of the source database data to which the refreshable clone is refreshed.



The timestamp can be a minimum of 1 minute in the past and a maximum of 7 days in the past. The timestamp must be later than, that is after, the last refresh point. Thus, if you create a refreshable clone, then you would need to wait some time before you refresh the clone.

3. Click **Refresh clone**.

On the Oracle Cloud Infrastructure console the Lifecycle state shows **Updating** until the refresh operation completes. While the database is updating connections and queries wait until the refresh completes.

You can see the status of the refresh operation under **Work requests**.

On the Oracle Cloud Infrastructure Console, under Clone Information the **Refresh point** shows the timestamp of the source database's data that the clone was refreshed to.

> **Note:**
>
> Refreshable clones have a one week refresh age limit. If you do not perform a refresh within a week then the refreshable clone will not be refreshable. The refresh by date is shown in the Oracle Cloud Infrastructure Console banner. In the case where a refreshable clone is not refreshable, you can disconnect the database from the source to make it a read/write (standard) database. See Disconnect a Refreshable Clone from the Source Database for more information.

Disconnect a Refreshable Clone from the Source Database

Shows you the steps to disconnect a refreshable clone from the source database.

When you disconnect a refreshable clone the refreshable clone is disassociated from the source database. This converts the database from a refreshable clone to a regular database. Following the disconnect operation you are allowed to reconnect the disconnected database to the source database. The reconnect operation is limited to a 24 hour period.

There are several ways to find refreshable clones. See View Refreshable Clones for an Autonomous Database Instance for more information.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

Starting from the Source Database Console Disconnect a refreshable clone from the source database and convert the refreshable clone to a read/write database:

1. On the **Details** page, under **Resources** select **Refreshable clones**.

   This shows the list of refreshable clones.

2. In the row for the refreshable clone you want to disconnect, click  at the end of the row and select **Disconnect clone from source database**.
3. In the Disconnect refreshable clone dialog enter the source database name to confirm disconnecting the clone.
4. Click **Disconnect refreshable clone**.

Starting from the Clone Database Console Disconnect a refreshable clone from the source database and convert the refreshable clone to a read/write database:

1. On the Details page, under Clone information, in the **Clone source** field click the **Disconnect** link.
2. In the Disconnect refreshable clone dialog enter the source database name to confirm disconnecting the clone.
3. Click **Disconnect refreshable clone**.

The Autonomous Database Lifecycle state of the cloned database changes to **Updating**. When the disconnect operation completes the Lifecycle state changes to **Available** and the Mode shows **Read/Write**.

After you disconnect a refreshable clone, the Oracle Cloud Infrastructure Console updates with the following changes:

- The **Display name** on the Autonomous Databases page updates and does not show the **Refreshable clone** indicator for the cloned instance.
- On the disconnected instance, the Autonomous Database Details page updates and does not show the **Refreshable clone** indicator, and the **Clone information** area is removed.
- On the disconnected instance, the Autonomous Database details page displays a banner indicating the date and time up to which you can reconnect the database to the source. The banner also includes a **Reconnect refreshable clone** button. For example:

See Reconnect a Refreshable Clone to the Source Database for more information.

- On the source database, on the **Autonomous Database Details** page, when you click **Refreshable clones** under **Resources**, the list no longer shows an entry for the disconnected clone.

Notes for disconnecting a refreshable clone.

- Disconnecting a refreshable clone from the source may fail if the source database has scaled down its storage and the refreshable clone has a larger amount of data than the source. In this case, you have the following options:
  - You may scale up the storage on the source database temporarily before disconnecting, and then scale the source back down after disconnecting.
  - Refresh the clone to a point where the scale down of storage occurred.
- After you disconnect a refreshable clone you have 24 hours to reconnect. After the reconnect period is over, the reconnect operation is not available. If you do not reconnect a disconnected refreshable clone, the clone is a standard Autonomous Database and there is no longer an option to reconnect the database to the source database.
- A disconnected refreshable clone is no longer associated with the source database. To use the database or to initiate the reconnect operation, you must know the name of the refreshable clone database that was disconnected from the source database. You must initiate the reconnect operation from the disconnected clone database. You cannot reconnect a disconnected clone from the source database.

Reconnect a Refreshable Clone to the Source Database

Shows you the steps to reconnect a disconnected clone to the source database.

After you disconnect a refreshable clone, during the following 24 hour reconnect period you can reconnect to the source database. The reconnect operation restores all the data back to that time when the clone was disconnected.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

Reconnect a database and restore all the data back to that time when the refreshable clone was disconnected from the source database as follows:

1. On the **Details** page, from the **More actions** drop-down list, select **Reconnect refreshable clone**.

   This shows the Reconnect Refreshable Clone dialog.
2. In the Reconnect Refreshable Clone dialog, enter the source database name to confirm.

> **Note:**
>
> All data and metadata inserted, updated, or deleted from the database where you run the reconnect operation will be lost when the database is reconnected to the source database.

3. Click **Reconnect refreshable clone**.

The Autonomous Database Lifecycle state changes to **Updating**.

When the reconnect refreshable clone operation completes, the Oracle Cloud Infrastructure Console on the clone indicates the change as follows:

- The Mode shows **Read Only**.
- If the clone was available when you initiated the reconnect operation, the Lifecycle state changes to **Available**.
- If the clone was stopped when you initiated the reconnect operation, the Lifecycle state shows **Stopped**.
- On the Autonomous Database Details page, the Clone Information area is updated. The **Clone type** shows Refreshable Clone, the **Clone source** shows the source database link, the **Refresh** link, **Disconnect** link, and the **Refresh point** shows the last refresh timestamp.

Notes for reconnecting a disconnected clone to the source database:

- If you disconnect a refreshable clone, you have 24 hours to use the reconnect operation. After the reconnect period is over, the reconnect operation is not available. If you do not reconnect a disconnected refreshable clone, the Autonomous Database is a standard read/write database and there is no longer an option to reconnect the database to the source database.
- Network ACLs and Private Endpoint network configuration options on a refreshable clone can be changed when the clone is disconnected from the source database. When you reconnect a refreshable clone to its source database, the reconnect operation does not restore the network ACLs or Private Endpoint network configuration options on the refreshable clone.

Create a Cross Tenancy or Cross-Region Refreshable Clone

You can create a refreshable clone in a different tenancy (destination tenancy). When you create a cross tenancy refreshable clone, the refreshable clone in the destination tenancy can be in the same region or in a different region (cross-region).

The cross tenancy cloning option is only available using the CLI or the Autonomous Database REST APIs. This option is not available using the Oracle Cloud Infrastructure Console.

These steps cover creating a Refreshable clone. See Create a Cross Tenancy or Cross-Region Clone for details on creating a cross tenancy Full clone or Metadata clone.

To create a cross tenancy refreshable clone:

1. Perform the prerequisite steps to define the OCI Identity and Access Management policies to authorize cross tenancy cloning.

   See Prerequisites for Cross Tenancy Cloning for details.

> **Note:**
>
> If the polices you specify to allow cross tenancy cloning are subsequently revoked, cross-tenancy cloning is no longer allowed. In addition, for a refreshable clone, when the policies are revoked actions on the remote tenancy that require contact with the source tenancy, such as disconnect and refresh will fail.

2. On the tenancy where you want to create the refreshable clone, use the CLI or call the REST API.

   You run the CLI or call the REST API on the destination tenancy in the destination region where you want to create the refreshable clone. The source database resides in a different tenancy, and if specified in a different region. This creates either a cross-tenancy refreshable clone in the same region, or a cross-tenancy refreshable clone in a different region (cross-region).

   For example, with the CLI:

   ```
   oci db autonomous-database create-refreshable-clone
       --compartment-id ocid1.tenancy.oc1..aaaaaaaafcue47pqmrf4vigne_example
       --source-id ocid1.autonomousdatabase.oc1.iad.anuwcljs4bv3yyiae2_example
       --db-name adatabasedb1
       --data-storage-size-in-tbs 1
       --compute-model ECPU
       --compute-count 2
       --refreshable-mode MANUAL
   ```

   See create-refreshable-clone for more information.

   Use the `CreateAutonomousDatabase` API to create a cross tenancy refreshable clone. When you call this API, to create a refreshable clone you set the `source` value to `CLONE_TO_REFRESHABLE` and the `sourceID` to the OCID of the source database.

   See the following for information on the REST API:

   - CreateAutonomousDatabase
   - CreateRefreshableAutonomousDatabaseCloneDetails

- For information about using the API and signing requests, see REST APIs and Security Credentials.
- For information about SDKs, see Software Development Kits and Command Line Interface.

Use the API

Provides details for using the APIs to create, refresh, list, disconnect, and reconnect a refreshable clone.

For information about using the API and signing requests, see REST APIs and Security Credentials. For information about SDKs, see Software Development Kits and Command Line Interface.

Use these API operations to manage a refreshable clone:

- To create a refreshable clone, use CreateAutonomousDatabase.
- To refresh a refreshable clone, use AutonomousDatabaseManualRefresh.

- To list the attached refreshable clones for a database, use ListAutonomousDatabaseClones.
- To disconnect a refreshable clone from the source database, use UpdateAutonomousDatabase.

Refreshable Clone Notes

Lists limitations and notes for Autonomous Database refreshable clones.

- Always Free Autonomous Databases do not support refreshable clones.
- Autonomous Database does not support using customer-managed encryption keys with a refreshable clone. You cannot create a refreshable clone from a source database that uses customer-managed encryption keys. Additionally, you cannot switch to a customer-managed encryption key on a source database that has one or more refreshable clones.
- You cannot create a cascading series of refreshable clones. Thus, a refreshable clone cannot be created from another refreshable clone.
- You cannot backup or restore a refreshable clone.
- The ADMIN password for a refreshable clone is inherited from the source database. If you want to change the ADMIN password for a refreshable clone you must change the ADMIN password on the source database, and then refresh the clone for the ADMIN password on the clone to come into effect.
- Oracle APEX URLs do not work in a refreshable clone read-only database and the APEX URLs are disabled in the Oracle Cloud Infrastructure Console and in the Database Actions APEX and APEX Workspaces cards. APEX URLs are enabled for a read/write database when a refreshable clone is disconnected from the source.
- Oracle Machine Learning is disabled in a refreshable clone read-only database. The OML User Administration URLs are disabled in the Oracle Cloud Infrastructure Console and on the Database Actions Launchpad.
- Oracle Data Safe is not supported for a refreshable clone instance. The Data Safe data in the refreshable clone's source database, for example user registration, data masking, and so on are available and Data Safe can be enabled when a refreshable clone is disconnected from the source.
- When you scale up or scale down the storage on the source Autonomous Database instance for a refreshable clone, the change is immediately reflected in the database console and in the billing for the refreshable clone. When the refreshable clone is refreshed to a refresh point after the scale up or down operation, the system makes a corresponding change to the refreshable clone storage (scaling up or scaling down the refreshable clone storage to match the source database).
- You cannot use the rename operation on a refreshable clone instance or on a database that is the source for a refreshable clone.
- When the patch level of a source Autonomous Database instance is **Regular**, in regions that support the **Early** patch level you can set the patch level of a clone to **Early**. See Set the Patch Level for more information.
- When the patch level of a source Autonomous Database instance is **Early**, you can only choose the **Early** patch level for your clone (the refreshable clone must use the same **Early** patch level). See Set the Patch Level for more information.
- Automatic Workload Repository (AWR) data and reports are not available for refreshable clones. In addition, the graphs that rely on AWR data are not available, including the following graphs:

  - The Running SQL statements graph on the Overview tab shown on the **Database Dashboard** card in Database Actions.
  - The SQL statement response time graph on the Overview tab shown on the **Database Dashboard** card in Database Actions.
  - The Time period graphs on the Monitor tab shown on the **Database Dashboard** card in Database Actions.
  - The Performance Hub graph data older than one hour is not available.
- For an Oracle Autonomous JSON Database (workload type **JSON Database**), note the following when reconnecting to the source database:
  - If, after you disconnect the refreshable clone, you promote both the clone and the source to Oracle Autonomous Transaction Processing (workload type **Transaction Processing**), you can reconnect the database to the source.
  - If after you disconnect the refreshable clone, you promote the source database to Oracle Autonomous Transaction Processing (workload type **Transaction Processing**) and do not promote the disconnected clone, the disconnected clone must also be promoted to Oracle Autonomous Transaction Processing (workload type **Transaction Processing**) before you perform the reconnect operation.
  - If after you disconnect the refreshable clone, you promote the disconnected database to Oracle Autonomous Transaction Processing (workload type **Transaction Processing**), you can still reconnect to the source but the reconnected database remains in the promoted state.

  See Promote to Autonomous Transaction Processing for more information.

Manage Autonomous Database Built-in Tools

Autonomous Database includes built-in tools that you can enable and disable when you provision or clone a database, or at any time for an existing database.

- About Autonomous Database Built-in Tools
  Provides a complete list of Autonomous Database built-in tools and a description of each tool.
- View Autonomous Database Built-in Tools Status
  Provides the steps to view the status of built-in tools on your Autonomous Database instance.
- Configure Autonomous Database Built-in Tools
  You can enable or disable Autonomous Database built-in tools for an existing Autonomous Database instance. In addition, with the ECPU compute model you can configure limits for the number of ECPUs and the idle time allowed for several of the built-in tools.
- Configure Autonomous Database Built-in Tools when you Provision or Clone an Instance
  During provisioning or when you clone an instance, you can enable or disable Autonomous Database built-in tools.
- Notes for Autonomous Database Built-in Tools
  Provides additional Autonomous Database built-in tools notes and lists differences that apply depending on the workload type of your Autonomous Database instance.

About Autonomous Database Built-in Tools

Provides a complete list of Autonomous Database built-in tools and a description of each tool.

Tool	Description
Oracle APEX	Oracle APEX is a low-code development platform that you can use to build scalable, secure enterprise applications that can be deployed anywhere. See Create Applications with Oracle APEX in Autonomous Database for more information.
Database Actions	Oracle Database Actions is a web-based interface that uses Oracle REST Data Services to provide development, data tools, administration and monitoring features for Oracle Autonomous Database. See Connect with Built-In Oracle Database Actions for more information.
Graph Studio	Graph Studio automates the creation of knowledge (RDF) and property graphs and includes interactive tooling for query, analysis, and visualization of these graphs in the Autonomous Database. You must log in as a graph-enabled user to access Graph Studio. Create this user in Database Actions. See Using Oracle Graph with Autonomous Database for more information.
Oracle ML User Interface	Oracle Machine Learning User Interface provides immediate access to the Oracle Machine Learning components and functionality on Autonomous Database, including OML Notebooks, OML AutoML UI, OML Models, and template example notebook. This includes: • OML Notebooks • OML AutoML UI • OML for Python • OML for R • OML Services
Data Transforms	Oracle Data Transforms allows you to design graphical data transformations in the form of data flows and workflows. The data flows define how the data is moved and transformed between different systems, while the workflows define the sequence in which the data flows run.
Web Access (ORDS)	Oracle REST Data Services (ORDS) provides HTTPS interfaces for working with the contents of your Oracle Database in one or more REST enabled schema See Develop with Oracle REST Data Services on Autonomous Database for more information.
MongoDB API	Oracle Database API for MongoDB enables MongoDB-compatible clients and drivers to connect directly to Autonomous Database. See Using Oracle Database API for MongoDB for more information.
SODA Drivers	Simple Oracle Document Access (SODA) is a set of APIs that let you work with JSON documents managed by the Oracle Database without needing to use SQL. SODA drivers are available for REST, Java, Node.js, Python, PL/SQL, and C. See Use SODA for REST with Autonomous Database for more information.

- About Configuring Built-in Tools Compute Resource Limits

About Configuring Built-in Tools Compute Resource Limits

When your Autonomous Database instance is using the ECPU compute model, some built-in tools have compute resource limits. Optionally you can configure the compute resource limits for these tools.

For several built-in tools, the **ECPU count** and **Maximum idle time** configuration options let you specify resource allocation for the VMs that run the associated built-in tool. When you configure a built-in tool, the values for these configuration options mean the following in terms of resource usage and billing:

- You do not pay for a tool's ECPU allocation if you do not use the tool.
- The VMs associated with a tool are provisioned when you start using a tool. For example, if Graph Studio is disabled, billing does not begin when you enable the tool. Billing begins when you start using the Graph Studio.
- The **ECPU count** specifies the CPU allocation that is dedicated to the tool. The value you enter for **ECPU count** applies in addition to the database **ECPU count** you specify for the instance. After you start using a tool with a specified **ECPU count**, you are billed per ECPU hour reserved from the time the built-in Tool is launched.
- Billing stops for a built-in tool's allocated ECPUs when the built-in tool is disabled, the instance stops or is terminated, or when the built-in tool is idle for more than the specified **Maximum idle time**.

  The default maximum idle times depend on the tool:

Built-in Tool	Default Maximum Idle Time (Minutes)
Graph Studio	240
Oracle Machine Learning User Interface	60
Data Transforms	10

View Autonomous Database Built-in Tools Status

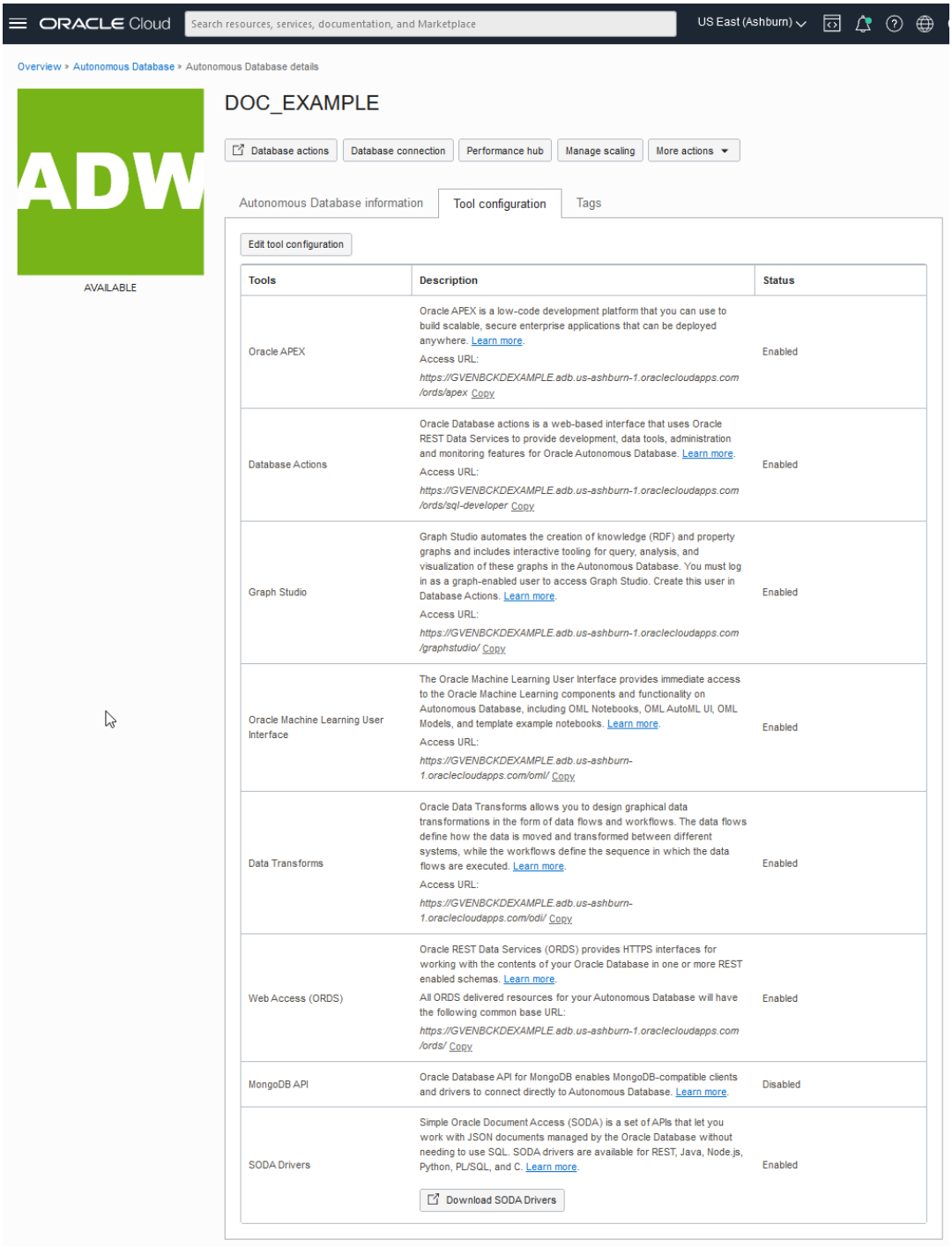
Provides the steps to view the status of built-in tools on your Autonomous Database instance.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

1. Navigate to the Autonomous Database details page.
2. Select the **Tool configuration** tab.

   This displays the tools configuration details and status.



Configure Autonomous Database Built-in Tools

You can enable or disable Autonomous Database built-in tools for an existing Autonomous Database instance. In addition, with the ECPU compute model you can configure limits for the number of ECPUs and the idle time allowed for several of the built-in tools.

- Configure Autonomous Database Built-in Tools (ECPU compute model)
- Configure Autonomous Database Built-in Tools (OCPU compute model)

Configure Autonomous Database Built-in Tools (ECPU compute model)

Describes how to enable or disable Autonomous Database built-in tools for an existing Autonomous Database instance.

Tool status is retained across starts and restarts for an Autonomous Database instance. For example, if a tool is disabled before you stop or restart an instance, the tool maintains the same status (disabled) after a start or restart.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

1. On the Autonomous Database details page, select the **Tool configuration** tab.
2. Click **Edit tool configuration**.
3. Select or deselect in the **Enable tool** field to enable or disable a tool.
4. (Optional) When enabled, Graph Studio, Oracle Machine Learning User Interface, and Data Transforms provide default **ECPU count** and **Maximum idle count** values and let you configure these values.

   Change the **ECPU count** value if you want to provide more or less resources for a built-in tool.

   Change the **Maximum idle time** value to specify the time, in minutes, after which an idle Graph Studio, Data Transforms, or Oracle Machine Learning VM is terminated.

   See About Configuring Built-in Tools Compute Resource Limits for more information.
5. Click **Apply** to apply your changes.

Configure Tools

Tools	Description	Enable tool	Configuration
Oracle APEX	Oracle APEX is a low-code development platform that you can use to build scalable, secure enterprise applications that can be deployed anywhere. Learn more.	Enabled	--
Database Actions	Oracle Database actions is a web-based interface that uses Oracle REST Data Services to provide development, data tools, administration and monitoring features for Oracle Autonomous Database. Learn more.	Enabled	--
Graph Studio	Graph Studio automates the creation of knowledge (RDF) and property graphs and includes interactive tooling for query, analysis, and visualization of these graphs in the Autonomous Database. You must log in as a graph-enabled user to access Graph Studio. Create this user in Database Actions. Learn more.	Enabled	ECPU count: 2 Enable ECPUs, between 2 and 256, for Graph Studio. Maximum idle time (Minutes): 240 This is the time, in minutes, after which an idle Graph Studio VM will be terminated.
Oracle Machine Learning User Interface	The Oracle Machine Learning User Interface provides immediate access to the Oracle Machine Learning components and functionality on Autonomous Database, including OML Notebooks, OML AutoML UI, OML Models, and template example notebooks. Learn more.	Enabled	ECPU count: 2 Enable ECPUs, between 2 and 16, for Oracle Machine Learning. Maximum idle time (Minutes): 60 This is the time, in minutes, after which an idle Oracle Machine Learning VM will be terminated.
Data Transforms	Oracle Data Transforms allows you to design graphical data transformations in the form of data flows and workflows. The data flows define how the data is moved and transformed between different systems, while the workflows define	Enabled	ECPU count: 2 Enable ECPUs, between 2 and 16, for Data Transforms. Maximum idle time (Minutes): 10 This is the time, in minutes, after which an

The **Lifecycle State** changes to **Updating**. When the request completes, the **Lifecycle State** shows **Available**.

Configure Autonomous Database Built-in Tools (OCPU compute model)

Describes how to enable or disable Autonomous Database built-in tools for an existing Autonomous Database instance.

Tool status is retained across starts and restarts for an Autonomous Database instance. For example, if a tool is disabled before you stop or restart an instance, the tool maintains the same status (disabled) after a start or restart.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

1. On the Autonomous Database details page, select the **Tool configuration** tab.
2. Click **Edit tool configuration**.
3. Select or deselect in the **Enable tool** field to enable or disable a tool.
4. Click **Apply** to apply your changes.

Configure Tools

Tools	Description	Enable tool
Oracle APEX	Oracle APEX is a low-code development platform that you can use to build scalable, secure enterprise applications that can be deployed anywhere. Learn more.	Enabled
Database Actions	Oracle Database actions is a web-based interface that uses Oracle REST Data Services to provide development, data tools, administration and monitoring features for Oracle Autonomous Database. Learn more.	Enabled
Graph Studio	Graph Studio automates the creation of knowledge (RDF) and property graphs and includes interactive tooling for query, analysis, and visualization of these graphs in the Autonomous Database. You must log in as a graph-enabled user to access Graph Studio. Create this user in Database Actions. Learn more.	Enabled
Oracle Machine Learning User Interface	The Oracle Machine Learning User Interface provides immediate access to the Oracle Machine Learning components and functionality on Autonomous Database, including OML Notebooks, OML AutoML UI, OML Models, and template example notebooks. Learn more.	Enabled
Data Transforms	Oracle Data Transforms allows you to design graphical data transformations in the form of data flows and workflows. The data flows define how the data is moved and transformed between different systems, while the workflows define the sequence in which the data flows are executed. Learn more.	Enabled
Web Access (ORDS)	Oracle REST Data Services (ORDS) provides HTTPS interfaces for working with the contents of your Oracle Database in one or more REST enabled schemas. Learn more.	Enabled
MongoDB API	Oracle Database API for MongoDB enables MongoDB-compatible clients and drivers to connect directly to Autonomous Database. Learn more.	Disabled Use Access Control List or Private Endpoint as your network access type to be able to enable MongoDB API

Apply Cancel

The **Lifecycle State** changes to **Updating**. When the request completes, the **Lifecycle State** shows **Available**.

Configure Autonomous Database Built-in Tools when you Provision or Clone an Instance

During provisioning or when you clone an instance, you can enable or disable Autonomous Database built-in tools.

- Configure Autonomous Database Built-in Tools when you Provision or Clone an Instance (ECPU compute model)
- Configure Autonomous Database Built-in Tools when you Provision or Clone an Instance (OCPU compute model)

Configure Autonomous Database Built-in Tools when you Provision or Clone an Instance (ECPU compute model)

Describes how to enable or disable Autonomous Database built-in tools when you provision or clone an instance.

When you are cloning, note the following:

- When you are cloning and you select **Full Clone** or **Metadata Clone**, the built-in tools default settings are based on the defaults for the workload type of the clone. In this case, the defaults are not based on source database built-in tools settings.
- When you are cloning and you select **Refreshable Clone**, the refreshable clone inherits the built-in tool status from the source database. Editing the built-in tool status for refreshable clone is not be possible during or after provisioning. If the source database of refreshable clone changes its tool configuration, this is reflected in the refreshable clone after the next refresh.

Perform the steps to provision or clone an instance:

- Provision Autonomous Database
- Clone an Autonomous Database Instance

1. Click **Show advanced options** to select advanced options.
2. In advanced options, select the **Tools** tab.
3. Click **Edit tool configuration**.
4. Select or deselect a tool in the **Enable tool** field to enable or disable the tool.
5. (Optional) When enabled, Graph Studio, Oracle Machine Learning User Interface, and Data Transforms provide default **ECPU count** and **Maximum idle count** values and let you configure these values.

   Change the **ECPU count** value if you want to provide more or less resources for a built-in tool.

   Change the **Maximum idle time value** to specify the time, in minutes, after which an idle Graph Studio, Data Transforms, or Oracle Machine Learning VM is terminated.

   See About Configuring Built-in Tools Compute Resource Limits for more information.
6. Click **Apply** to apply the configuration changes and return to the provision or clone steps.

Configure Tools

Tools	Description	Enable tool	Configuration
Oracle APEX	Oracle APEX is a low-code development platform that you can use to build scalable, secure enterprise applications that can be deployed anywhere. Learn more.	Enabled	--
Database Actions	Oracle Database actions is a web-based interface that uses Oracle REST Data Services to provide development, data tools, administration and monitoring features for Oracle Autonomous Database. Learn more.	Enabled	--
Graph Studio	Graph Studio automates the creation of knowledge (RDF) and property graphs and includes interactive tooling for query, analysis, and visualization of these graphs in the Autonomous Database. You must log in as a graph-enabled user to access Graph Studio. Create this user in Database Actions. Learn more.	Enabled	ECPU count: 2 Enable ECPUs, between 2 and 256, for Graph Studio. Maximum idle time (Minutes): 240 This is the time, in minutes, after which an idle Graph Studio VM will be terminated.
Oracle Machine Learning User Interface	The Oracle Machine Learning User Interface provides immediate access to the Oracle Machine Learning components and functionality on Autonomous Database, including OML Notebooks, OML AutoML UI, OML Models, and template example notebooks. Learn more.	Enabled	ECPU count: 2 Enable ECPUs, between 2 and 16, for Oracle Machine Learning. Maximum idle time (Minutes): 60 This is the time, in minutes, after which an idle Oracle Machine Learning VM will be terminated.
Data Transforms	Oracle Data Transforms allows you to design graphical data transformations in the form of data flows and workflows. The data flows define how the data is moved and transformed between different systems, while the workflows define	Enabled	ECPU count: 2 Enable ECPUs, between 2 and 16, for Data Transforms. Maximum idle time (Minutes): 10 This is the time, in minutes, after which an

Configure Autonomous Database Built-in Tools when you Provision or Clone an Instance (OCPU compute model)

Describes how to enable or disable Autonomous Database built-in tools when you provision or clone an instance

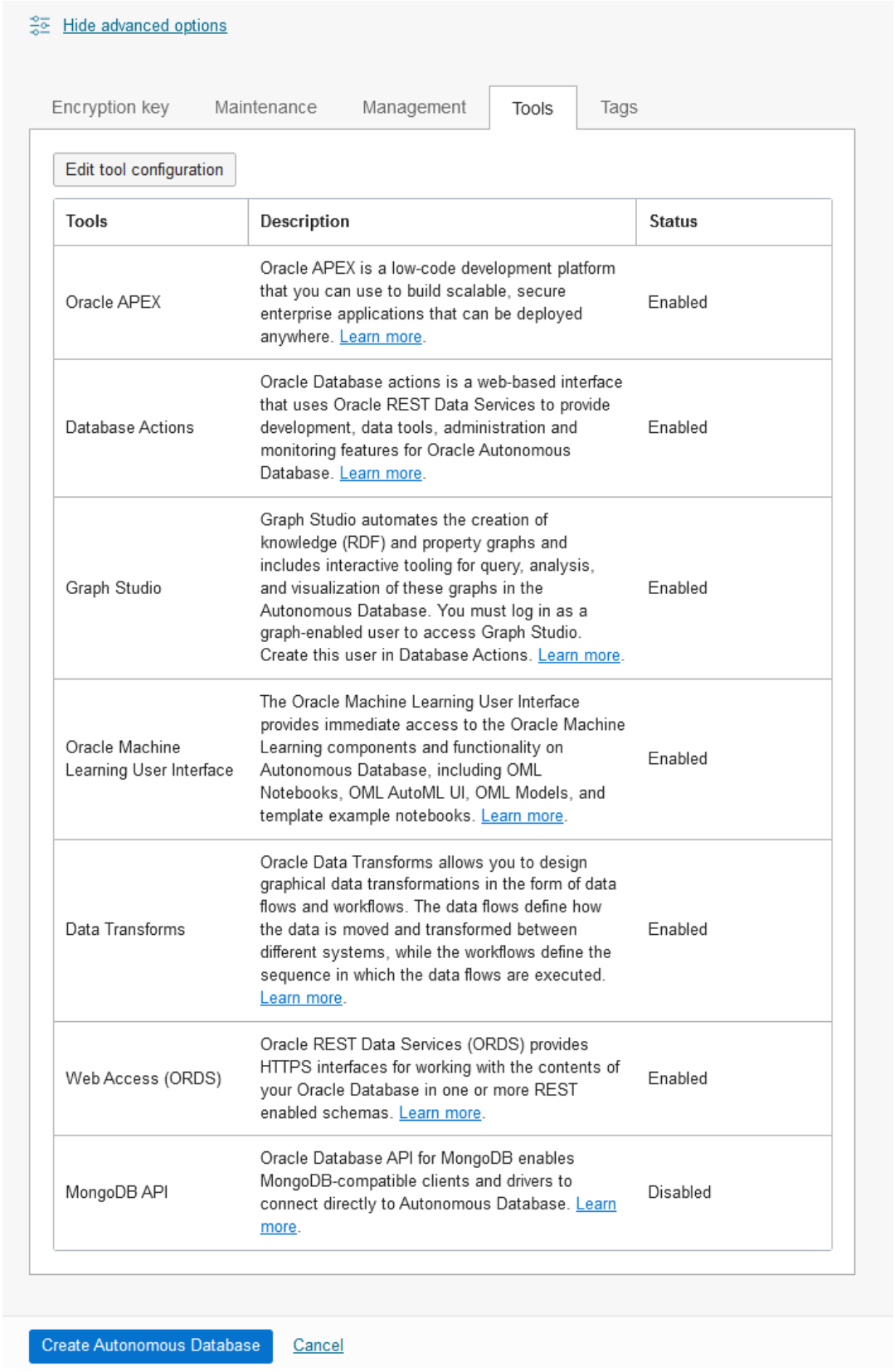
When you are cloning, note the following:

- When you are cloning and you select **Full Clone** or **Metadata Clone**, the built-in tools default settings are based on the defaults for the workload type of the clone. In this case, the defaults are not based on source database built-in tools settings.
- When you are cloning and you select **Refreshable Clone**, the refreshable clone inherits the built-in tool status from the source database. Editing the built-in tool status for refreshable clone is not be possible during or after provisioning. If the source database of refreshable clone changes its tool configuration, this is reflected in the refreshable clone after the next refresh.

Perform the steps to provision or clone an instance:

- Provision Autonomous Database
- Clone an Autonomous Database Instance

1. Click **Show advanced options** to select advanced options.
2. In advanced options, select the **Tools** tab.
3. Click **Edit tool configuration**.
4. Select or deselect a tool in the **Enable tool** field to enable or disable the tool.
5. Click **Apply** to apply the configuration changes and return to the provision or clone steps.


Hide advanced options
Encryption key
Maintenance
Management
Tools
Tags
Edit tool configuration
Tools
Description
Status
Oracle APEX
Oracle APEX is a low-code development platform that you can use to build scalable, secure enterprise applications that can be deployed anywhere. Learn more.
Enabled
Database Actions
Oracle Database actions is a web-based interface that uses Oracle REST Data Services to provide development, data tools, administration and monitoring features for Oracle Autonomous Database. Learn more.
Enabled
Graph Studio
Graph Studio automates the creation of knowledge (RDF) and property graphs and includes interactive tooling for query, analysis, and visualization of these graphs in the Autonomous Database. You must log in as a graph-enabled user to access Graph Studio. Create this user in Database Actions. Learn more.
Enabled
Oracle Machine Learning User Interface
The Oracle Machine Learning User Interface provides immediate access to the Oracle Machine Learning components and functionality on Autonomous Database, including OML Notebooks, OML AutoML UI, OML Models, and template example notebooks. Learn more.
Enabled
Data Transforms
Oracle Data Transforms allows you to design graphical data transformations in the form of data flows and workflows. The data flows define how the data is moved and transformed between different systems, while the workflows define the sequence in which the data flows are executed. Learn more.
Enabled
Web Access (ORDS)
Oracle REST Data Services (ORDS) provides HTTPS interfaces for working with the contents of your Oracle Database in one or more REST enabled schemas. Learn more.
Enabled
MongoDB API
Oracle Database API for MongoDB enables MongoDB-compatible clients and drivers to connect directly to Autonomous Database. Learn more.
Disabled
Create Autonomous Database
Cancel

Notes for Autonomous Database Built-in Tools

Provides additional Autonomous Database built-in tools notes and lists differences that apply depending on the workload type of your Autonomous Database instance.

- Disabling ORDS affects built-in tools such as Database Actions, APEX, and MongoDB API. Therefore, if you disable ORDS, this also disables APEX, Database Actions, and MongoDB API.
- When ORDS is disabled, Database Actions, APEX, and MongoDB API are disabled. If you subsequently enable ORDS this does not automatically enable APEX, Database Actions, or the MongoDb APIs. You must manually enable each built-in tool that is dependent on ORDS. Enable ORDS first, then you can enable APEX, Database Actions, or the MongoDB APIs
- Always Free Autonomous Database does not provide configuration options for Autonomous Database tools and does not allow you to enable or disable Autonomous Database tools.
- By default when you provision an Autonomous Database with Autonomous Data Warehouse or Autonomous Transaction Processing workload types, the MongoDB API is disabled. If MongoDB API is disabled, you can enable it during provisioning or cloning, or in an existing database.

  You can enable MongoDB API only if you have an ACL or if you are on a private endpoint. See Configure Access for MongoDB for more information.
- For JSON Database, Graph Studio is not supported and enabling or disabling Graph Studio for JSON Database is not possible.

  For JSON Database, MongoDB API is enabled by default and you can disable it during the provisioning or cloning, or in an existing database.
- When you are cloning and you select **Full Clone** or **Metadata Clone**, the built-in tools default settings are based on the defaults for the workload type of the clone. In this case, the defaults are not based on source database built-in tools settings.
- When you are cloning and you select **Refreshable Clone**, the refreshable clone inherits the built-in tool status from the source database. Editing the built-in tool status for refreshable clone is not be possible during or after provisioning. If the source database of refreshable clone changes its tool configuration, this is reflected in the refreshable clone after the next refresh.
- When you provision or clone an Autonomous Database with the APEX workload type (APEX service), the only supported built-in tools are Oracle APEX, Web Access (ORDS), and Database Actions. In the APEX service, you can only disable Database Actions.
- All existing sessions end after you disable a built-in tool. In addition, disabling Oracle APEX or Web Access (ORDS) also affects the existing applications that are based on these tools. For example, a user application based on ORDS does not continue to work after you disable ORDS.

Use Autonomous Database Events

Oracle Cloud Infrastructure Events enable you to create automation based on state changes for your Autonomous Database.

You can use Oracle Cloud Infrastructure Events to subscribe to and be notified of Autonomous Database events. The Oracle Cloud Infrastructure Events allow you to create automation and to receive notifications based on state changes for resources.

About Events Based Notification and Automation on Autonomous Database

You can subscribe to Oracle Cloud Infrastructure Events.

An event could be a resource lifecycle state change or a system event impacting a resource. For example, an event is emitted on Autonomous Database when a backup or restore operation begins or ends. Subscribing to events allows you to receive notifications or to perform other types of automation for events.

You can subscribe to Autonomous Database events as follows:

- Critical Events: See Critical Events on Autonomous Database for details on the events in the Critical event category.
- Information Events: See Information Events on Autonomous Database for details on the events in the Information event category.
- Individual Events. See Autonomous Database Event Types for the list of Autonomous Database events.
- Autonomous Data Guard Events. See Events and Notifications for a Standby Database for the list of Autonomous Data Guard events.

See Overview of Events for complete information on Oracle Cloud Infrastructure Events.

See Get Notified of Autonomous Database Events for information on creating event rules.

Critical Events on Autonomous Database

Critical events on Autonomous Database are issues that cause disruption to the database.

Event	Description
`AdminPasswordWarning`	Provides a message that the Autonomous Database ADMIN password is expiring within 30 days or is expired. • If the ADMIN password is within 30 days of being unusable, you receive a warning indicating the date when the ADMIN password can no longer be used. • If the ADMIN password expires and is no longer usable, Autonomous Database reports an event that specifies that the ADMIN user password has expired and must be reset.
`AutomaticFailoverBegin`	When an automatic failover begins for the Autonomous Database and the database is unavailable an `AutonomousDatabase-AutomaticFailoverBegin` event is generated. The event will only be triggered if you are using Autonomous Data Guard.

Event	Description
`AutomaticFailoverEnd`	When an automatic failover that was triggered is complete and the database is now available an `AutonomousDatabase-AutomaticFailoverEnd` event is generated. The event will only be triggered if you are using Autonomous Data Guard.
`DatabaseDownBegin`	The Autonomous Database instance cannot be opened, or the services such as high, low, medium, tp, or tpurgent are not started or available. The following conditions do not trigger `DatabaseDownBegin`: • Operations performed during the maintenance window • Load balancer, network, or backup related issues • A user stopping the instance This event will not be triggered if you are using Autonomous Data Guard and the standby database is not available due to any of these conditions.
`DatabaseDownEnd`	The database is recovered from the down state, meaning the Autonomous Database instance is opened with its services, following a `DatabaseDownBegin` event. `DatabaseDownEnd` is triggered only if there was a preceding `DatabaseDownBegin` event. The following conditions do not trigger `DatabaseDownEnd`: • Operations performed during the maintenance window • A user starting the instance If you are using Autonomous Data Guard and the primary database goes down, this triggers a `DatabaseDownBegin` event. If the system fails over to the standby database, this triggers a `DatabaseDownEnd` event.
`DatabaseInaccessibleBegin`	The Autonomous Database instance is using customer-managed keys and the database becomes inaccessible (state shows **Inaccessible**). The following conditions trigger `DatabaseInaccessibleBegin`: • Oracle Cloud Infrastructure Vault master encryption key is deleted. • Oracle Cloud Infrastructure Vault master encryption key is disabled. • Autonomous Database instance is not able to reach the Oracle Cloud Infrastructure Vault.
`DatabaseInaccessibleEnd`	If the Autonomous Database instance recovers from the **Inaccessible** state that generated a `DatabaseInaccessibleBegin` event, when the database state changes to **Available**, the database triggers a `DatabaseInaccessibleEnd` event.
`FailedLoginWarning`	If the number of failed login attempts reaches 3 * *number of total users* in the last three (3) hours, the database triggers a `FailedLoginWarning` event. Failed login attempts could be due to a user entering an invalid username or password, or due to a connection timeout. The event is triggered when the failed login attempts threshold is exceeded within the monitored time period (the last three hours).

Event	Description
InstanceRelocateBegin	If an Autonomous Database is relocated to another Exadata infrastructure due to server maintenance, hardware refresh, a hardware issue, or as part of a resource scale up, the InstanceRelocateBegin event is triggered at the start of the relocation process.
InstanceRelocateEnd	If an Autonomous Database is relocated to another Exadata infrastructure due to server maintenance, hardware refresh, a hardware issue, or as part of a resource scale up, the InstanceRelocateEnd event is triggered at the end of the relocation process.
WalletExpirationWarning	This event is generated when Autonomous Database determines that a wallet is due to expire in less than six (6) weeks. This event is reported at most once per week. This event is triggered when there is a connection that uses the wallet that is due to expire.

Use the event type **Autonomous Database - Critical** to specify critical events in Event rules.

Information Events on Autonomous Database

Information events provide important details about database lifecycle events, workload capture and reply events, and some other non-critical events on Autonomous Database.
The times shown with information events are in UTC.

Event	Description
AJDNonJsonStorageExceeded	This event is generated when an Autonomous JSON Database has exceeded the maximum storage limit of 20GB of data stored outside of SODA collections. This limit does not apply to data stored in SODA collections or objects associated with SODA collections, such as indexes or materialized views. In addition to this event, an email is sent to the account owner. You must either reduce your usage of non-SODA-related data to below the 20GB limit or promote the Autonomous JSON Database to Autonomous Transaction Processing. See Promote to Autonomous Transaction Processing for more information.
APEXUpgradeAvailable	This event is generated when you are using Oracle APEX and a new release becomes available.
APEXUpgradeBegin	This event is generated when you are using Oracle APEX and your Autonomous Database instance begins an upgrade to a new Oracle APEX release or begins to apply an Oracle APEX Patch Set Bundle.
APEXUpgradeEnd	This event is generated when you are using Oracle APEX and your Autonomous Database instance completes an upgrade to a new Oracle APEX release or completes the installation of an Oracle APEX Patch Set Bundle.
DatabaseConnection	This event is generated if a connection is made to database from a new IP address (the connection has not been made from the specified IP address in the last 30 days).

Event	Description
`InactiveConnectionsDetected`	This event is generated when the number of inactive database connections detected is above a certain ratio compared to all database connections for the Autonomous Database instance. Subscribing to this event can help you keep track of unused connections. This event is generated once per day, when the following is true: • The number of inactive connections, where the elapsed time for the inactive connection is greater than 24 hours, is great than 10% of the total number of connections in any state. For this calculation, inactive connections are connections where the `status` is `"INACTIVE"`. Use the following query to report detailed information on sessions that have been inactive for more than 24 hours: `SELECT sid, serial#, last_call_et FROM v$session WHERE status = 'INACTIVE' AND last_call_et > 60 * 60 *24;`
`Long term backup ended`	This event is triggered when a long term backup completes.
`Long term backup schedule disabled`	This event is triggered when a long term backup schedule is disabled.
`Long term backup schedule enabled / updated`	This event is triggered when a long term backup schedule is either enabled or updated.
`Long term backup started`	This event is triggered when a long term backup starts.
`MaintenanceBegin`	This event is triggered when the maintenance starts and provides the start timestamp for the maintenance (this event does not provide the scheduled start time).
`MaintenanceEnd`	This event is triggered when the maintenance ends and provides the end timestamp for the maintenance (this event does not provide the scheduled end time).
`NewMaintenanceSchedule`	This event is generated when the maintenance date is updated and the new date is shown on the Oracle Cloud Infrastructure Console. When the Java version will be updated, the event `description` field indicates this with the following: `This maintenance involves Java version update, please expect down time of Java service during this maintenance window.`
`OperatorAccess`	This event is triggered when operator access is detected for the database. You can query the access details from the `DBA_OPERATOR_ACCESS` view with the request ID provided in the event description. The `OperatorAccess` event is generated once every 24 hours.
`ScheduledMaintenanceWarning`	This event is generated when the instance is 24 hours from a scheduled maintenance and again when the instance is 1 hour (60 minutes) from the scheduled maintenance. When the Java version will be updated, the event `description` field indicates this with the following message: `This maintenance involves Java version update, please expect down time of Java service during this maintenance window.`
`WorkloadCaptureBegin`	This event is triggered when a workload capture is initiated.

Event	Description
`WorkloadCaptureEnd`	This event is triggered when a workload capture completes successfully. This generates a pre-authenticated (PAR) URL to download the capture reports. This URL is contained in the `captureDownloadURL` field of the event and is valid for 7 days from the date of generation.
`WorkloadReplayBegin`	This event is triggered when a workload replay is initiated.
`WorkloadReplayEnd`	This event is triggered when a workload replay completes successfully. This generates a pre-authenticated (PAR) URL to download the replay reports. This URL is contained in the `replayDownloadURL` field of the event and is valid for 7 days from the date of generation.

> **Note:**
>
> When Autonomous Data Guard is enabled, any of these events that occur on the standby database do not trigger an Information event.

Use the event type **Autonomous Database - Information** to specify information events in Event rules.

See View Patch and Maintenance Window Information, Set the Patch Level for information on maintenance windows.

Autonomous Database Event Types

Provides a list of Autonomous Database events.

Friendly Name	Event Type
Autonomous Database - Access Control Lists Update Begin	`com.oraclecloud.databaseservice.updateautonomousdatabaseacl.begin`
Autonomous Database - Access Control Lists Update End	`com.oraclecloud.databaseservice.updateautonomousdatabaseacl.end`
Autonomous Database - Auto Scaling Disabled	`com.oraclecloud.databaseservice.autonomousdatabaseautoscaledisabled`
Autonomous Database - Auto Scaling Enabled	`com.oraclecloud.databaseservice.autonomousdatabaseautoscaleenabled`
Autonomous Database - Automatic Backup Begin	`com.oraclecloud.databaseservice.automaticbackupautonomousdatabase.begin`
Autonomous Database - Automatic Backup End	`com.oraclecloud.databaseservice.automaticbackupautonomousdatabase.end`
Autonomous Database - Automatic Refresh Begin The "Automatic Refresh" events are used for Autonomous Database refreshable clones.	`com.oraclecloud.databaseservice.automaticrefresh.begin`
Autonomous Database - Automatic Refresh End	`com.oraclecloud.databaseservice.automaticrefresh.end`

Friendly Name	Event Type
Autonomous Database - Automatic Refresh Failed	`com.oraclecloud.databaseservice.automaticrefresh.failed`
Autonomous Database - Change Compartment Begin	`com.oraclecloud.databaseservice.changeautonomousdatabasecompartment.begin`
Autonomous Database - Change Compartment End	`com.oraclecloud.databaseservice.changeautonomousdatabasecompartment.end`
Autonomous Database - Change Database Name Begin	`com.oraclecloud.databaseservice.changeautonomousdatabasename.begin`
Autonomous Database - Change Database Name End	`com.oraclecloud.databaseservice.changeautonomousdatabasename.end`
Autonomous Database - Create Backup Begin	`com.oraclecloud.databaseservice.autonomous.database.backup.begin`
Autonomous Database - Create Backup End	`com.oraclecloud.databaseservice.autonomous.database.backup.end`
Autonomous Database - Create Begin	`com.oraclecloud.databaseservice.autonomous.database.instance.create.begin`
Autonomous Database - Create End	`com.oraclecloud.databaseservice.autonomous.database.instance.create.end`
Autonomous Database - Critical See Critical Events on Autonomous Database for more information.	`com.oraclecloud.databaseservice.autonomous.database.critical`
Autonomous Database - Deregister Autonomous Database with Data Safe Begin	`com.oraclecloud.databaseservice.deregisterautonomousdatabasedatasafe.begin`
Autonomous Database - Deregister Autonomous Database with Data Safe End	`com.oraclecloud.databaseservice.deregisterautonomousdatabasedatasafe.end`
Autonomous Database - Disable Data Guard Begin	`com.oraclecloud.databaseservice.disableautonomousdataguard.begin`
Autonomous Database - Disable Data Guard End	`com.oraclecloud.databaseservice.disableautonomousdataguard.end`
Autonomous Database - Disconnect Refreshable Clone from Source Database Begin	`com.oraclecloud.databaseservice.disconnectrefreshableautonomousdatabaseclonefromsource.begin`
Autonomous Database - Disconnect Refreshable Clone from Source Database End	`com.oraclecloud.databaseservice.disconnectrefreshableautonomousdatabaseclonefromsource.end`
Autonomous Database - Enable Data Guard Begin	`com.oraclecloud.databaseservice.enableautonomousdataguard.begin`
Autonomous Database - Enable Data Guard End	`com.oraclecloud.databaseservice.enableautonomousdataguard.end`
Autonomous Database - Free Database Automatic Stop Reminder This event is emitted 48 hours prior to database stop.	`com.oraclecloud.databaseservice.freeautonomousdatabasestopreminder`
Autonomous Database - Free Database Automatically Stopped	`com.oraclecloud.databaseservice.freeautonomousdatabasestopped`
Autonomous Database - Free Database Automatic Termination Reminder. This event is emitted 48 hours prior to database termination.	`com.oraclecloud.databaseservice.freeautonomousdatabaseterminationreminder`

Friendly Name	Event Type
Autonomous Database - Free Database Automatically Terminated	com.oraclecloud.databaseservice.freeautonomousdatabaseterminated
Autonomous Database - Generate Wallet	com.oraclecloud.databaseservice.generateautonomousdatabasewallet
Autonomous Database - Information See Information Events on Autonomous Database for more information.	com.oraclecloud.databaseservice.autonomous.database.information
Autonomous Database - Manual Failover Begin	com.oraclecloud.databaseservice.failoverautonomousdatabase.begin
Autonomous Database - Manual Failover End This event is emitted after failover completes successfully or unsuccessfully. Unsuccessful manual failovers may result in data loss. Check the Autonomous Database details page in the Oracle Cloud Infrastructure Console for additional information in the event of an unsuccessful manual failover.)	com.oraclecloud.databaseservice.failoverautonomousdatabase.end
Autonomous Database - Manual Refresh Begin The "Manual Refresh" events are used for Autonomous Database refreshable clones.	com.oraclecloud.databaseservice.manualrefresh.begin
Autonomous Database - Manual Refresh End	com.oraclecloud.databaseservice.manualrefresh.end
Autonomous Database - Deregister Autonomous Database with Data Safe Begin	com.oraclecloud.databaseservice.deregisterautonomousdatabasedatasafe.begin
Autonomous Database - Deregister Autonomous Database with Data Safe End	com.oraclecloud.databaseservice.deregisterautonomousdatabasedatasafe.end
Autonomous Database - Register Autonomous Database with Data Safe Begin	com.oraclecloud.databaseservice.registerautonomousdatabasedatasafe.begin
Autonomous Database - Register Autonomous Database with Data Safe End	com.oraclecloud.databaseservice.registerautonomousdatabasedatasafe.end
Autonomous Database - Restart Begin	com.oraclecloud.databaseservice.restartautonomousdatabase.begin
Autonomous Database - Restart End	com.oraclecloud.databaseservice.restartautonomousdatabase.end
Autonomous Database - Restore Begin	com.oraclecloud.databaseservice.autonomous.database.restore.begin
Autonomous Database - Restore End	com.oraclecloud.databaseservice.autonomous.database.restore.end
Autonomous Database - Rotate Encryption Key Begin	com.oraclecloud.databaseservice.rotateautonomousdatabaseencryptionkey.begin
Autonomous Database - Rotate Encryption Key End	com.oraclecloud.databaseservice.rotateautonomousdatabaseencryptionkey.end
Autonomous Database - Start Begin	com.oraclecloud.databaseservice.startautonomousdatabase.begin
Autonomous Database - Start End	com.oraclecloud.databaseservice.startautonomousdatabase.end
Autonomous Database - Stop Begin	com.oraclecloud.databaseservice.stopautonomousdatabase.begin

Friendly Name	Event Type
Autonomous Database - Stop End	`com.oraclecloud.databaseservice.stopautonomousdatabase.end`
Autonomous Database - Switchover Begin	`com.oraclecloud.databaseservice.switchoverautonomousdatabase.begin`
Autonomous Database - Switchover End This event is emitted after switchover completes successfully or unsuccessfully. Unsuccessful switchovers may result in data loss. Check the Autonomous Database details page in the Oracle Cloud Infrastructure Console for additional information in the event of an unsuccessful switchover.	`com.oraclecloud.databaseservice.switchoverautonomousdatabase.end`
Autonomous Database - Terminate Begin	`com.oraclecloud.databaseservice.deleteautonomousdatabase.begin`
Autonomous Database - Terminate End	`com.oraclecloud.databaseservice.deleteautonomousdatabase.end`
Autonomous Database - Update Begin	`com.oraclecloud.databaseservice.updateautonomousdatabase.begin`
Autonomous Database - Update End	`com.oraclecloud.databaseservice.updateautonomousdatabase.end`
Autonomous Database - Update Open Mode Begin	`com.oraclecloud.databaseservice.updateautonomousdatabaseopenmode.begin`
Autonomous Database - Update Open Mode End	`com.oraclecloud.databaseservice.updateautonomousdatabaseopenmode.end`
Autonomous Database - Upgrade Database Version Begin	`com.oraclecloud.databaseservice.upgradeautonomousdatabasedbversion.begin`
Autonomous Database - Upgrade Database Version End	`com.oraclecloud.databaseservice.upgradeautonomousdatabasedbversion.end`

Get Notified of Autonomous Database Events

Using Oracle Cloud Infrastructure Events you can subscribe to Autonomous Database events individually or in two categories, Critical and Information events.

> **Note:**
>
> When you subscribe to an event category, either Critical or Information, you are notified when any events in the category occur. For example, to get notified when the database goes down, subscribe to the Autonomous Database Critical event type.

If you want to subscribe to Critical or Information events:

- To subscribe to Critical events, create an event rule using the **Autonomous Database - Critical** event type.
- To subscribe to Information events, create an event rule using the **Autonomous Database - Information** event type.

If you want to subscribe to one or more individual Critical or Information events, create an event rule and add conditions. For example, to create an event rule that only applies for the `AdminPasswordWarning` Critical event, create a rule as follows:

1. On the Oracle Cloud Infrastructure Console click **Observability & Management**.
2. Under **Events Service**, click **Rules**.
3. To add a rule, click **Create Rule**.
4. On the Create Rule page, enter a Display Name and a Description.
5. Under Rule Conditions, enter a rule condition
   a. Enter the Condition: **Event Type**.
   b. Enter the Service Name: **Database**.
   c. Enter the Event Type, one of: **Autonomous Database - Critical** or **Autonomous Database - Information**.
6. Click **+ Another Condition**.
7. Under Condition, select **Attribute**.
   a. Under Attribute Name, select **eventName**.
   b. Under Attribute Values, enter the event name. For example, enter `AdminPasswordWarning`.

      See Critical Events on Autonomous Database and Information Events on Autonomous Database for the list of event names.

   For example with these entries the Rule Logic area shows:

```
MATCH event WHERE (
   eventType EQUALS ANY OF (
      com.oraclecloud.databaseservice.autonomous.database.critical
   )
   AND (
         eventName MATCHES ANY OF (
             AdminPasswordWarning
         )
   )
)
```

8. In the **Actions** area, select the actions you want.
9. Click **Create Rule**.

See Getting Started with Events for information on using Oracle Cloud Infrastructure Events, creating event rules, and for information on configuring actions for notifications.

Manage Time Zone File Updates on Autonomous Database

Autonomous Database provides several options to automatically update time zone files on an Autonomous Database database instance.

- About Time Zone File Update Options
  For time zone support Oracle Database uses time zone files that store the list of all time zones. The time zone files for Autonomous Database are periodically updated to reflect the latest time zone specific changes.
- Use AUTO_DST_UPGRADE Time Zone File Option

- Use AUTO_DST_UPGRADE_EXCL_DATA Time Zone File Option

About Time Zone File Update Options

For time zone support Oracle Database uses time zone files that store the list of all time zones. The time zone files for Autonomous Database are periodically updated to reflect the latest time zone specific changes.

Autonomous Database provides the following options for updating time zone files:

- `AUTO_DST_UPGRADE`: Automatically upgrades the time zone files and automatically updates the data on your database to use the latest time zone data. This option requires that you restart or stop and then start your Autonomous Database instance.
- `AUTO_DST_UPGRADE_EXCL_DATA`: Automatically upgrades the time zone files and does not automatically update the data on your database to use the latest time zone data. The time zone files are upgraded to the latest version when you enable this option and versions are kept up to date with the latest version. This option does not require that you restart your Autonomous Database instance.

> **Note:**
>
> By default, both `AUTO_DST_UPGRADE` and `AUTO_DST_UPGRADE_EXCL_DATA` are disabled. You can enable one or the other of these options, but not both.

With every Daylight Saving Time (DST) version release, there are DST file changes introduced to make existing data comply with the latest DST rules. Applying a change to the database time zone files not only affects the way new data is handled, but potentially alters data stored in `TIMESTAMP WITH TIME ZONE` columns.

When a load or import operation results in the following time zone related error, this indicates that the version of your time zone files is out of date and you need to update to the latest version available for your database:

```
ORA-39405: Oracle Data Pump does not support importing from a source database
with TSTZ version y
into a target database with TSTZ version n.
```

You can enable the `AUTO_DST_UPGRADE` feature that automatically upgrades an instance to use the latest available version of the time zone files, and automatically updates the rows on your database to use the latest time zone data. The latest version of time zone files are applied and values in `TIMESTAMP WITH TIME ZONE` columns are updated at the next database restart. However, depending on your database usage, the required database restart might restrict you from using `AUTO_DST_UPGRADE` to upgrade to the latest time zone files.

You can enable `AUTO_DST_UPGRADE_EXCL_DATA` to update your time zone files. This option allows you to immediately update your database if you encounter import/export errors due to a version mismatch of time zone files (where you receive the `ORA-3940` error).

Enabling `AUTO_DST_UPGRADE_EXCL_DATA` on your database can be beneficial in the following cases:

- Data on the instance is in UTC and the database is not impacted by DST time zone file updates.

- The instance does not have any data in `TIMESTAMP WITH TIME ZONE` columns.
- Data in the instance uses dates that are not altered with new time zone or daylight saving time policies.

In this case, when your Autonomous Database instance does not contain rows that are adversely impacted by the new time zone rules and you encounter the `ORA-3940` error, you can enable the `AUTO_DST_UPGRADE_EXCL_DATA` option to update to the latest version of the time zone files. The `AUTO_DST_UPGRADE_EXCL_DATA` option prioritizes the success of Oracle Data Pump jobs over the data consistency issues due to changing DST.

See Oracle Support Document 406410.1 to help you determine whether time zone changes will affect your database.

> **Note:**
>
> Oracle recommends enabling the `AUTO_DST_UPGRADE` option when these limiting cases do not apply to your database.

In summary, the choice of enabling automatic time zone upgrades with `AUTO_DST_UPGRADE` or `AUTO_DST_UPGRADE_EXCL_DATA` involves the following considerations:

- Possible Data Inconsistency: Oracle recommends that you maintain your database on the latest time zone file version by enabling `AUTO_DST_UPGRADE`. This option automatically updates the rows on your database to use the latest time zone data.

  The alternative option, `AUTO_DST_UPGRADE_EXCL_DATA` allows you to maintain your database using the latest time zone version without requiring a database restart; however, data that might be affected by the updated time zone files is not updated, which could lead to possible data inconsistency.
- Database Restart: Oracle recommends that you maintain your database on the latest time zone file version by enabling `AUTO_DST_UPGRADE`. This option requires a restart to upgrade to the latest time zone files version and during the restart updates the rows for existing data of `TIMESTAMP WITH TIME ZONE` data type to use the latest version. Enabling `AUTO_DST_UPGRADE` can affect database restart time. A restart can require extra time compared to a restart without this option enabled when there are new time zone files and there is data that needs to be updated when the database restarts.

  When you enable `AUTO_DST_UPGRADE_EXCL_DATA`, at the time you enable this option the database is upgraded to the latest version the time zone files and during each maintenance window, if newer time zone files are available, Autonomous Database updates database to use the latest version. Enabling `AUTO_DST_UPGRADE_EXCL_DATA` does not require that you restart your Autonomous Database instance to keep up to date with the latest version of the time zone files.

See Datetime Data Types and Time Zone Support for more information.

Use AUTO_DST_UPGRADE Time Zone File Option

Autonomous Database provides the `AUTO_DST_UPGRADE` option to automatically update time zone files on an Autonomous Database database instance.

The `AUTO_DST_UPGRADE` feature automatically upgrades the time zone files and automatically upgrades the rows on your database to use the latest time zone data.

> **Note:**
>
> By default, both `AUTO_DST_UPGRADE` and `AUTO_DST_UPGRADE_EXCL_DATA` are disabled. You can enable one or the other of the automatic time zone file update options, but not both.

When you enable `AUTO_DST_UPGRADE` your Autonomous Database instance automatically applies updates for time zone files, depending on the state of the instance:

- **Stopped**: At the next start operation the update is automatically applied.
- **Available**: After a restart, or a stop and then start, the update is automatically applied.

When a load or import operation results in the following time zone related error, this indicates that your time zone files are out of date and you need to update to the latest version available for your database:

```
ORA-39405: Oracle Data Pump does not support importing from a source database
with TSTZ version n+1
into a target database with TSTZ version n.
```

To enable automatic time zone file updates with `AUTO_DST_UPGRADE`:

1. Enable the `AUTO_DST_UPGRADE` feature.

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_FEATURE(
        feature_name => 'AUTO_DST_UPGRADE');
END;
/
```

2. Query `dba_cloud_config` to verify that `AUTO_DST_UPGRADE` is enabled.

```
SELECT param_name, param_value FROM dba_cloud_config WHERE
       LOWER(param_name) = 'auto_dst_upgrade';

PARAM_NAME              PARAM_VALUE
----------------------- ----------------------------------------
auto_dst_upgrade        enable
```

3. Query `dba_cloud_config` to check the time zone version.

```
SELECT param_name, param_value FROM dba_cloud_config
    WHERE LOWER(param_name) = 'latest_timezone_version';

PARAM_NAME              PARAM_VALUE
----------------------- -----------
latest_timezone_version 38
```

You can query the DB_NOTIFICATIONS view to see if the time zone version is the latest version:

```
SELECT type, time, description, expected_start_date FROM db_notifications
   WHERE TYPE='TIMEZONE VERSION';
```

When AUTO DST UPGRADE is enabled, Autonomous Database upgrades to the latest version of the time zone files (when you next restart, or stop and then start the database). The columns in your database with the datatype TIMESTAMP WITH TIME ZONE are converted to the new time zone version during the restart.

To disable AUTO_DST_UPGRADE:

1. Disable the AUTO_DST_UPGRADE feature:

```
BEGIN
   DBMS_CLOUD_ADMIN.DISABLE_FEATURE(
        feature_name => 'AUTO_DST_UPGRADE');
END;
/
```

2. Query dba_cloud_config to verify that AUTO_DST_UPGRADE is disabled.

```
SELECT param_name, param_value FROM dba_cloud_config WHERE
       LOWER(param_name) = 'auto_dst_upgrade';

0 rows selected.
```

See ENABLE_FEATURE Procedure and DISABLE_FEATURE Procedure for more information.

See TIMESTAMP WITH TIME ZONE Data Type for more information.

Use AUTO_DST_UPGRADE_EXCL_DATA Time Zone File Option

Autonomous Database provides the AUTO_DST_UPGRADE_EXCL_DATA option to automatically update time zone files on an Autonomous Database database instance.

The AUTO_DST_UPGRADE_EXCL_DATA automatically upgrades the time zone files and does not automatically upgrade the rows on your database to use the latest time zone data. When this option is enabled the database upgrades to the latest version of the time zone files and subsequently upgrades the database to use new versions of the time zone files during the Autonomous Database maintenance window (whenever a new version is available). This option does not require that you restart your Autonomous Database instance.

> **Note:**
>
> By default, both `AUTO_DST_UPGRADE` and `AUTO_DST_UPGRADE_EXCL_DATA` are disabled. You can enable one or the other of the automatic time zone file update options, but not both.

When a load or import operation results in the following time zone related error, this indicates that your time zone files are out of date and you need to update to the latest version available for your database:

```
ORA-39405: Oracle Data Pump does not support importing from a source database
with TSTZ version n+1
into a target database with TSTZ version n.
```

To enable automatic time zone file updates with `AUTO_DST_UPGRADE_EXCL_DATA`:

1. Enable the `AUTO_DST_UPGRADE_EXCL_DATA` feature.

   ```
   BEGIN
      DBMS_CLOUD_ADMIN.ENABLE_FEATURE(
           feature_name => 'AUTO_DST_UPGRADE_EXCL_DATA');
   END;
   /
   ```

2. Query `dba_cloud_config` to verify that `AUTO_DST_UPGRADE_EXCL_DATA` is enabled.

   ```
   SELECT param_name, param_value FROM dba_cloud_config WHERE
          LOWER(param_name) = 'auto_dst_upgrade_excl_data';

   PARAM_NAME                     PARAM_VALUE
   ------------------------------ ----------------------------------------
   auto_dst_upgrade_excl_data     enabled
   ```

3. Query `dba_cloud_config` to check the time zone version.

   ```
   SELECT param_name, param_value FROM dba_cloud_config
       WHERE LOWER(param_name) = 'latest_timezone_version';

   PARAM_NAME              PARAM_VALUE
   ----------------------- -----------
   latest_timezone_version 38
   ```

   You can query the `DB_NOTIFICATIONS` view to see if the time zone version is the latest version:

   ```
   SELECT type, time, description, expected_start_date FROM db_notifications
      WHERE TYPE='TIMEZONE VERSION';
   ```

When `AUTO_DST_UPGRADE_EXCL_DATA` is enabled, Autonomous Database upgrades to the latest version of the time zone files and checks and updates to new time zone file versions during each scheduled maintenance window. With `AUTO_DST_UPGRADE_EXCL_DATA` enabled, the

columns in your database with the datatype `TIMESTAMP WITH TIME ZONE` are not converted to the new time zone version.

To disable `AUTO_DST_UPGRADE_EXCL_DATA`:

1. Disable the `AUTO_DST_UPGRADE_EXCL_DATA` feature:

```
BEGIN
   DBMS_CLOUD_ADMIN.DISABLE_FEATURE(
        feature_name => 'AUTO_DST_UPGRADE_EXCL_DATA');
END;
/
```

2. Query `dba_cloud_config` to verify that `AUTO_DST_UPGRADE_EXCL_DATA` is disabled.

```
SELECT param_name, param_value FROM dba_cloud_config WHERE
       LOWER(param_name) = 'auto_dst_upgrade_excl_data';

0 rows selected.
```

See ENABLE_FEATURE Procedure and DISABLE_FEATURE Procedure for more information.

See TIMESTAMP WITH TIME ZONE Data Type for more information.

Monitor Autonomous Database Availability

Shows you the steps to view availability information for an Autonomous Database instance.

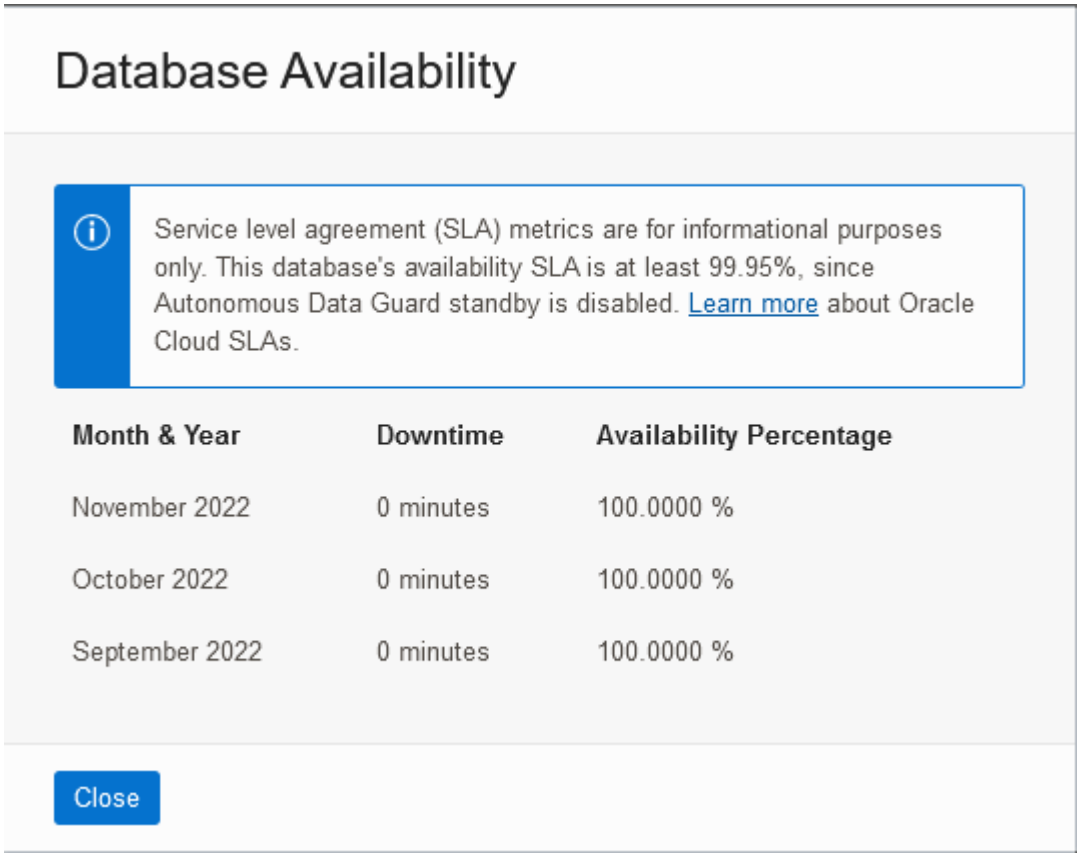
Database Availability is calculated based on the "Monthly Uptime Percentage" described in the Oracle PaaS and IaaS Public Cloud Services Pillar Document, document under **Delivery Policies** (see Autonomous Database Availability Service Level Agreement).

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To view Database Availability for an Autonomous Database instance:

1. On the **Details** page under General Information, click the Database Availability link in the **Lifecycle State** field.



2. Click **Close** to dismiss the page.

Note the following for the Autonomous Database instance availability:

- For databases with Cross-Region Autonomous Data Guard enabled, review both the primary database and the remote standby database availability.
- The Database Availability page shows information for the current month and for the previous two months.
- After you provision a new Autonomous Database instance, database availability information is only available for the current month.
- See View Metrics for an Autonomous Database Instance for more information on database availability. The Database Availability metric provides the Database Availability history for an Autonomous Database instance.
- Database availability information is not available prior to the introduction of this feature in April 2022.

See Monitor Regional Availability of Autonomous Databases for information on regional average availability metrics for Autonomous Database.

Monitor Regional Availability of Autonomous Databases

The regional availability is represented as a list of daily uptime percentages of Autonomous Databases across a set of services in each available region.

The regional availability data is updated daily on the following page:

Regional Average Availability Metrics for Oracle Autonomous Database

Notes for regional availability:

- The regional availability page shows information for the current month and for the previous two months.
- Regional availability information starts in June 2022 (availability data is not shown for dates prior to June 2022).

See View Metrics for an Autonomous Database Instance for more information regional availability.

See Monitor Autonomous Database Availability for information on Autonomous Database instance availability.

Managing and Viewing Maintenance Windows, Patching Options, Work Requests, and Customer Contacts

You can obtain information about system events, maintenance, and other important information for the operation of your Autonomous Database instance.

This includes the following:

- Oracle Cloud Infrastructure work requests allow you to monitor long-running operations such as cloning or backing up Autonomous Database.
- Oracle Cloud Infrastructure Console provides information on Autonomous Database maintenance windows and patching and provides a customer contacts area where you can view or manage the list of contacts to receive information on operational issues and service announcements.
- The `DBA_OPERATOR_ACCESS` view provides information on the top level SQL statements that Oracle Cloud Infrastructure cloud operations performs.

Topics

About Work Requests

You can view Oracle Cloud Infrastructure Work Requests on Autonomous Database.

A work request is an activity log that enables you to track each step in an operation's progress. Each work request has an Oracle Cloud Identifier (OCID) that allows you to interact with it programmatically.

Autonomous Database work requests are created for the following operations:

- Creating or terminating an Autonomous Database instance

- Starting or stopping an Autonomous Database instance
- Restoring an Autonomous Database instance
- Cloning an Autonomous Database instance
- Scaling database storage or CPU
- Updating the database license type
- Updating a database's network access control list (ACL)

Under **Resources** on the Oracle Cloud Infrastructure console click **Work Requests** to see recent work requests.

Click the link under **Operation** to see more details for a work request.

See Work Requests Integration for more information.

View Patch and Maintenance Window Information, Set the Patch Level

Autonomous Database uses predefined maintenance windows to automatically patch your database. You can view maintenance and patch information and see details for Autonomous Database maintenance history. When you provision your database you can select a patch level.

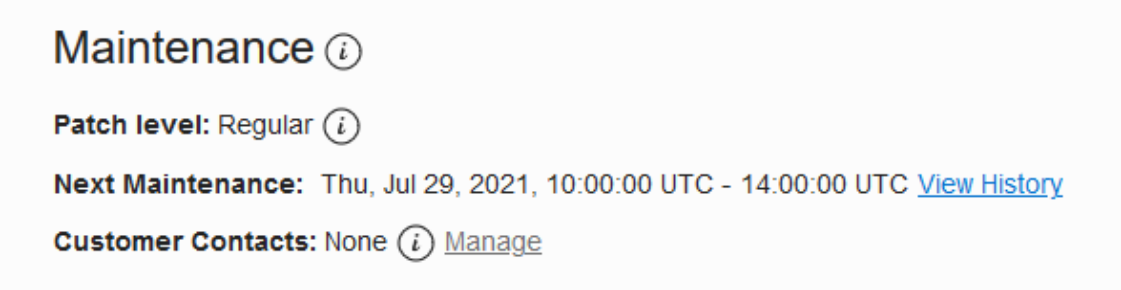
- About Scheduled Maintenance and Patching
  All Autonomous Database instances are automatically assigned to a maintenance window and different instances can have different maintenance windows.
- View Maintenance Event History
  You can view Autonomous Database maintenance event history for details about past maintenance events, such as the title, state, start time, and stop time.
- View Patch Details
  You can view Autonomous Database patch information, including a list of resolved issues and components.
- Set the Patch Level
  When you provision or clone an Autonomous Database instance you can select a patch level to apply to upcoming patches. There are two patch level options: **Regular** and **Early**.
- View Maintenance Status Notifications
  The `DB_NOTIFICATIONS` view stores information about maintenance status notifications for your Autonomous Database instance.

About Scheduled Maintenance and Patching

All Autonomous Database instances are automatically assigned to a maintenance window and different instances can have different maintenance windows.

The Autonomous Database Details page shows the **Next Maintenance** field that includes the date and time for the upcoming maintenance window; the date is updated automatically when the next maintenance window is scheduled. The **View History** link provides details on past maintenance. The **Patch Level** field shows the patch level setting for the Autonomous Database instance.

> **Note:**
>
> Your database remains available during the maintenance window. Your existing database connections may get disconnected briefly; however, you can immediately reconnect and continue using your database.



Notes for scheduled maintenance and patching:

- The Autonomous Database operations team never accesses your data unless you explicitly grant permission through a Service Request for a specified duration.
- If your database is in the stopped state during the maintenance window, the database changes from the patch are applied when you start your database.
- You can use Oracle Cloud Infrastructure Events to be notified when maintenance begins and ends. See Information Events on Autonomous Database for more information.
- If you want to change the assigned maintenance window to a different 2-hour window, file a Service Request at Oracle Cloud Support. If you want a specific time period for your maintenance window, you can request the time period with the same Service Request. If you request a specific time period for your maintenance window, the change can only be made if the time period you request is available for your database.
- If your database's allocated storage is 384 TB, you can choose a custom 2-hour window by filing a Service Request at Oracle Cloud Support (that is you can file a Service Request to request a specific day for your maintenance window).

View Maintenance Event History

You can view Autonomous Database maintenance event history for details about past maintenance events, such as the title, state, start time, and stop time.

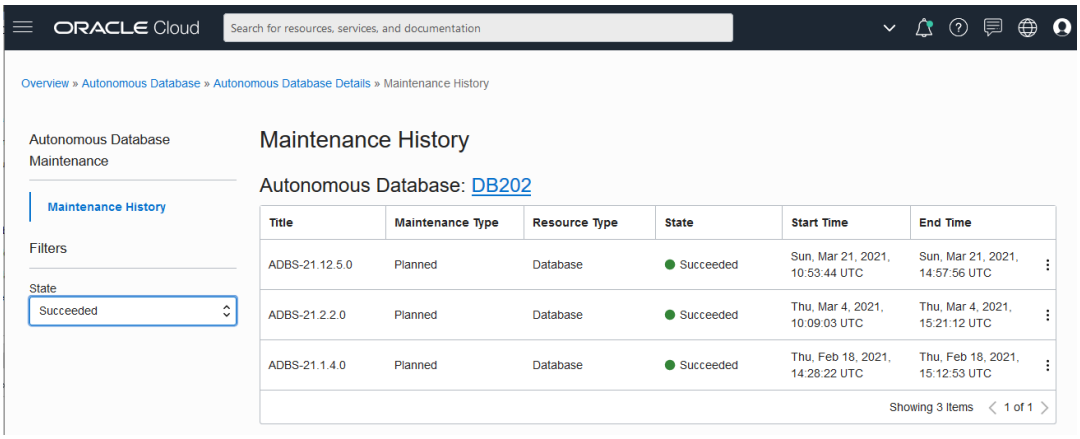
Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To view maintenance history, do the following:

1. On the Autonomous Database Details page, under **Maintenance**, click **View History**.
2. The Oracle Cloud Infrastructure Console displays the Maintenance History page.
3. (Optional) Use the **State** selector to filter events by state.

For example, if you select **Succeeded**, the Maintenance History page shows only the maintenance events with the Succeeded state.



The Maintenance History page shows details for each maintenance event, including the following:

- **Title**: The name of the maintenance event.
- **Maintenance Type**: Planned or Unplanned.
- **Resource Type**: The type of the resource on which the maintenance event occurs: Database or Infrastructure.
- **State**: Succeeded, Failed, or In progress.
- **Start Time**: Maintenance start time.
- **End Time**: Maintenance end time.

> **Note:**
>
> Maintenance event history is available starting with maintenance events after February 2021.

View Patch Details

You can view Autonomous Database patch information, including a list of resolved issues and components.

The `DBA_CLOUD_PATCH_INFO` view provides patch information related to reported bugs (bugs reported by a customer). You can use this information to determine if a bug you reported is fixed and the patch version where the fix was applied to your Autonomous Database instance. If there were no customer bugs in a patch, `DBA_CLOUD_PATCH_INFO` does not include any rows for that patch.

To view patch information for a specific patch, do the following:

1. Select the Autonomous Database patch that you want to view. The Maintenance History page on the Oracle Cloud Infrastructure Console shows the list of patches under **Title**.
2. For the selected patch, query the `DBA_CLOUD_PATCH_INFO` view.

For example, for patch `ADBS-21.7.1.1`, use the following query:

```
SELECT * FROM DBA_CLOUD_PATCH_INFO WHERE PATCH_VERSION = 'ADBS-21.7.1.1';
```

3. For an issue of interest, query the view to obtain details for the issue:

```
SELECT * FROM DBA_CLOUD_PATCH_INFO WHERE PATCH_VERSION = 'ADBS-21.7.1.1'
and BUG_NUM = bug_number;
```

To view patch information for all available patches:

```
SELECT * FROM DBA_CLOUD_PATCH_INFO;
```

Notes for viewing patch information:

- The view `DBA_CLOUD_PATCH_INFO` is available to the ADMIN user.
- Patch information and details on resolved issues is available from `ADBS-21.7.1.1` onwards (starting in July 2021).
- The view `DBA_CLOUD_PATCH_INFO` has the following columns:

```
BUG_NUM, BUG_TITLE, COMPONENT_NAME, PATCH_VERSION
```

See View Maintenance Status Notifications for details about the patches applied during maintenance.

Set the Patch Level

When you provision or clone an Autonomous Database instance you can select a patch level to apply to upcoming patches. There are two patch level options: **Regular** and **Early**.

When you select patch level **Early**, patches are applied for the Autonomous Database instance one week before the **Regular** scheduled patch. The **Next Maintenance** field in the Oracle Cloud Infrastructure Console reflects a maintenance window date and time based on the patch level.

The default patch level for provisioning an Autonomous Database instance is **Regular**. The default patch level for cloning is the patch level specified for the source database. Provisioning or cloning an instance and setting the patch level to **Early** allows you to use and to test upcoming patches before they are applied to all systems.

> **Note:**
>
> When cloning a source database with **Early** patch level, you can only choose **Early** patch level for your clone.

To set the patch level, do the following:

- Set the patch level when you provision or clone an instance.

  When you provision a new instance, follow the provisioning instructions and select the patch level, either **Regular** or **Early**. See Provision Autonomous Database for more information.

When you clone an instance, follow the cloning instructions and select a patch level, either **Regular** or **Early**. See Clone an Autonomous Database Instance for more information.

To change the patch level, do the following:

1. You cannot change the patch level for an existing Autonomous Database instance. The option to set the patch level is only available when you provision or clone an Autonomous Database instance.
2. Change the patch level by cloning a new instance and selecting a different patch level for the cloned database. Cloning a source database with the patch level **Regular** to **Early** is allowed. Cloning a source database with the patch level **Early** to **Regular** is not allowed. See Clone an Autonomous Database Instance for more information.

Reporting Patch Issues to Oracle Support

Oracle Support provides the same handling for regular or early patch level Autonomous Databases. If you are using an Autonomous Database instance and the patch level is **Early**, Oracle Support considers issues you report with high priority and after validating an issue, determines if the patch should be applied or withheld from the upcoming **Regular** patch.

If you have an issue to report, file a service request at Oracle Cloud Support or contact your support representative.

Notes for patching level:

- The option to set the patch level is not available in every region. In some regions all Autonomous Database instances are provisioned or cloned at the **Regular** patch level.
- Autonomous Data Guard is only available for instances with patch level **Regular**. When you configure an Autonomous Database instance with patch level **Early**, you cannot enable Autonomous Data Guard.
- Always Free Autonomous Database instances do not provide the **Early** patch level option.
- When the patch level of a source Autonomous Database instance is **Regular**, in regions that support the **Early** patch level you can set the patch level of a clone to **Early**.
- When the patch level of a source Autonomous Database instance is **Early**, you can only choose the **Early** patch level for your clone.

View Maintenance Status Notifications

The `DB_NOTIFICATIONS` view stores information about maintenance status notifications for your Autonomous Database instance.

To show notification information:

1. Connect to your Autonomous Database instance.
2. Use the following query to view maintenance (patching) information.

```
SELECT * FROM DB_NOTIFICATIONS WHERE TYPE = 'MAINTENANCE';
```

The following provides details about the `DESCRIPTION` field values.

- **Maintenance run has ended**: Specifies maintenance has completed. The `MAINTENANCE_STATUS` shows the value `COMPLETED` with the start and end timestamps for the completed maintenance in `ACTUAL_START_DATE` and `ACTUAL_END_DATE`.
- **Maintenance run is scheduled for the instance**: Specifies a new maintenance has been scheduled. The `MAINTENANCE_STATUS` shows the value `SCHEDULED` with the expected start

and end timestamps for the scheduled maintenance in `EXPECTED_START_DATE` and `EXPECTED_END_DATE`.

- **Maintenance run has begun**: Specifies the maintenance is in progress and provides the start timestamp for the active maintenance. The `MAINTENANCE_STATUS` shows the value `IN_PROGRESS` and `ACTUAL_START_DATE` stores the start timestamp.

The following table shows the `DB_NOTIFICATIONS` columns and datatypes.

Column	Datatype	Description
`TYPE`	`VARCHAR2(128)`	Specifies the type of the notification. Valid value is: `MAINTENANCE`.
`TIME`	`TIMESTAMP(6) WITH TIME ZONE`	Time when the notification entry was added.
`EXPECTED_START_DATE`	`TIMESTAMP(6) WITH TIME ZONE`	Scheduled maintenance start time.
`EXPECTED_END_DATE`	`TIMESTAMP(6) WITH TIME ZONE`	Scheduled maintenance end time.
`ACTUAL_START_DATE`	`TIMESTAMP(6) WITH TIME ZONE`	Actual maintenance start time.
`ACTUAL_END_DATE`	`TIMESTAMP(6) WITH TIME ZONE`	Actual maintenance end time.
`MAINTENANCE_PRODUCT`	`VARCHAR2(128)`	Product/component for which maintenance is scheduled/ongoing.
`MAINTENANCE_STATUS`	`VARCHAR2(128)`	Current status of the maintenance.
`DESCRIPTION`	`VARCHAR2(128)`	The notification message details.
`PATCH_ID`	`VARCHAR2(128)`	Patch version.

View and Manage Customer Contacts for Operational Issues and Announcements

You can view and manage the Autonomous Database customer contacts.

When customer contacts are set, Oracle sends notifications to the specified email addresses for Autonomous Database service-related issues. Contacts in the customer contacts list receive unplanned maintenance notices and other notices, including but not limited to notices for database upgrades and upcoming wallet expiration. When customer contacts are not set the notifications go to the tenancy admin email address associated with the account. Oracle recommends that you set the customer contacts so that the appropriate people receive service-related notifications.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To view or manage customer contact email addresses:

1. On the Autonomous Database Details page, under **Maintenance**, in the **Customer Contacts** field, click the **Manage** link.

This shows the Manage Contacts page. On the Manage Contacts page you can view the email contact list.

2. To add a contact, click **Add Contacts**.
3. Enter the contact email address.

   Oracle recommends adding an administrator group's email address over an individual's email address when possible, so as to not miss important notifications or announcements.
4. To add additional contacts, click **Add Contacts** again.
5. Click **Add Contacts** at the bottom of the page to add the new contacts.

The lifecycle state changes to Updating while the customer contact list is updated.

To remove customer contacts:

1. On the Manage Contacts page select the email addresses to remove.
2. To remove all email addresses, select the top column next to **Email**.
3. Click **Remove**.
4. On the Confirm remove contact page, click **Remove** to confirm.

The lifecycle state changes to Updating while the customer contact list is updated.

To edit customer contacts:

1. On the Manage Contacts page select the email addresses to edit.
2. To edit all email addresses, select the top column next to **Email**.
3. Click **Edit**.
4. On the Edit Contacts page edit the contacts as needed and click **Save**.

The lifecycle state changes to Updating while the customer contact list is updated.

Notes for adding customer contacts for operational issues and announcements:

- Oracle recommends adding an administrator group's email address over an individual's email address when possible, so as to not miss important notifications or announcements.
- In addition to sending information for operational issues and announcements to customer contacts, you can subscribe to Oracle Cloud Infrastructure Events to be notified of Autonomous Database events such as when regular maintenance is scheduled or when regular maintenance starts and ends. Event notification requires a subscription for each email address, independent of the customer contacts list.

  See Use Autonomous Database Events for more information.

View Oracle Cloud Infrastructure Operations Actions

The `DBA_OPERATOR_ACCESS` view stores information on the actions that Oracle Cloud Infrastructure cloud operations performs on your Autonomous Database.

These steps use **Database Actions** to query the `DBA_OPERATOR_ACCESS` view. You can query this view from any SQL connection by connecting as a user with the proper privileges.

1. On the Autonomous Database Details page select **Database Actions** and from the quick actions list click **SQL**.
2. Within the SQL Worksheet, query the `DBA_OPERATOR_ACCESS` view.

For example:

```
SELECT * FROM DBA_OPERATOR_ACCESS;
```

Notes for the `DBA_OPERATOR_ACCESS` view:

- You can continuously move this audit data to the object store for purposes such as integration with Security Information and Event Management (SIEM) tools. See Create and Configure a Pipeline for Export with Timestamp Column for more information on using pipelines to continuously export query results to object store.
- The `DBA_OPERATOR_ACCESS` view provides information starting on October 4, 2022, the date this feature was introduced. The view does not provide information prior to October 4, 2022.
- The `DBA_OPERATOR_ACCESS` view may not show results if Oracle Cloud Infrastructure cloud operations hasn't performed any actions or run any statements in your Autonomous Database instance. In this case, when you query the view you will not see any rows returned.
- DBA_OPERATOR_ACCESS View
  The `DBA_OPERATOR_ACCESS` view describes top level statements executed in the Autonomous Database instance by Oracle Cloud Infrastructure operations.

DBA_OPERATOR_ACCESS View

The `DBA_OPERATOR_ACCESS` view describes top level statements executed in the Autonomous Database instance by Oracle Cloud Infrastructure operations.

Column	Datatype	Description
`SQL_TEXT`	`VARCHAR2(4000)`	SQL text of the statement executed by the operator.
`EVENT_TIMESTAMP`	`DATE`	Timestamp of the operator action in UTC (Coordinated Universal Time).
`REQUEST_ID`	`VARCHAR2(64)`	Request number related to the reason behind the operator action. This could be a bug number, an SR number, or a change ticket request number that provides information on the reason for the action.
`REASON`	`VARCHAR2(64)`	Reason for the operator action. This provides context for the reason behind the action and may have a value such as: `MITIGATION`, `DIAGNOSTIC COLLECTION`, or `CUSTOMER REQUEST`.

Monitor and Manage Performance

Describes the Database Dashboard card in Database Actions where you can find information about the performance of an Autonomous Database instance.

- Monitor the Performance of Autonomous Database
  The **Database Dashboard** card in Database Actions provides the **Overview** and **Monitor** tabs where you can find information about the performance of an Autonomous Database instance.
- Monitor Autonomous Database with Performance Hub
  Use Performance Hub to monitor your database for a defined time period and download statistical reports. Performance Hub also lets you view real-time and historical performance data for an Autonomous Database instance.

- Manage Concurrency and Priorities on Autonomous Database
  Concurrency and prioritization of user requests in Autonomous Database is determined by the database service the user is connected with, and whether compute auto scaling is enabled.
- Manage CPU/IO Shares on Autonomous Database
  Autonomous Database comes with predefined CPU/IO shares assigned to different consumer groups. You can modify these predefined CPU/IO shares if your workload requires different CPU/IO resource allocations.
- Manage Runaway SQL Statements on Autonomous Database
  Specifies how you configure Autonomous Database to terminate SQL statements automatically based on their query runtime or their IO usage.
- Use Fast Ingest on Autonomous Database
- Monitor the Performance of Autonomous Database with Oracle Management Cloud
  Oracle Management Cloud allows you to monitor availability and performance for Autonomous Databases. You can use Oracle Database Management, part of Oracle Management Cloud, to monitor Autonomous Databases and On-premise Oracle Databases.
- Monitor Performance with Autonomous Database Metrics
  You can monitor the health, capacity, and performance of your databases with metrics, alarms, and notifications. You can use Oracle Cloud Infrastructure Console or Monitoring APIs to view metrics.
- Use Database Management Service to Monitor Databases
- Perform SQL Tracing on Autonomous Database
  Use SQL tracing to help you identify the source of an excessive database workload, such as a high load SQL statement in your application.
- Service Console Replacement with Database Actions
- Available Metrics: oci_autonomous_database
  This topic describes the metrics emitted by the Database service in the `oci_autonomous_database` namespace.

Monitor the Performance of Autonomous Database

The **Database Dashboard** card in Database Actions provides the **Overview** and **Monitor** tabs where you can find information about the performance of an Autonomous Database instance.

- Use Database Actions to Monitor Activity and Utilization
  Database Actions provides the **Database Dashboard** card, with **Overview** and **Monitor** tabs to provide real-time and historical information about the utilization of an Autonomous Database instance.
- Use Database Actions to Monitor Active Session History Analytics and SQL Statements
  In Database Actions the Performance Hub card provides information about Active Session History (ASH) analytics and current and past monitored SQL statements.

Use Database Actions to Monitor Activity and Utilization

Database Actions provides the **Database Dashboard** card, with **Overview** and **Monitor** tabs to provide real-time and historical information about the utilization of an Autonomous Database instance.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.

To view the **Overview** tab that shows general information about utilization, do the following:

1. Access Database Actions as the ADMIN user.

   See Access Database Actions as ADMIN for more information.
2. On the Database Actions Launchpad, under **Monitoring**, click **Database Dashboard**.

> **Note:**
>
> You can bookmark the Launchpad URL and go to that URL directly without logging in to the Oracle Cloud Infrastructure Console. If you logout and use the bookmark, then you need to enter the `ADMIN` username and *password*, and click **Sign in**. See Set the ADMIN Password in Autonomous Database if you need to change the password for the `ADMIN` user.

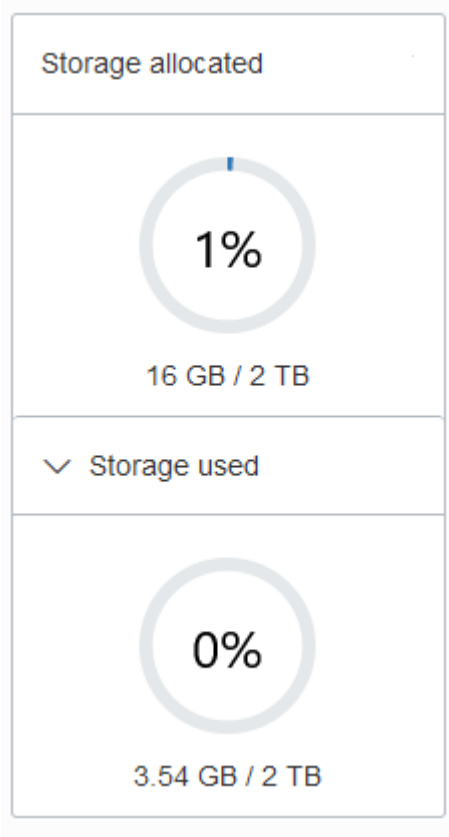
- Database Dashboard Overview
  The **Overview** tab shows real-time and historical information about the Autonomous Database utilization.
- Database Dashboard Activity
  The **Monitor** tab shows real-time and historical information about the Autonomous Database performance data, activity, and utilization.

Database Dashboard Overview

The **Overview** tab shows real-time and historical information about the Autonomous Database utilization.

The charts shown on this page include:

- **Storage:** This chart shows the provisioned, allocated, and used storage. The chart indicates what percentage of the space is currently in-use.



Provisioned storage is the amount of storage you select when you provision the instance or when you modify storage by scaling storage.

Storage allocated is the amount of storage physically allocated to all data tablespaces and temporary tablespaces and includes the free space in these tablespaces. This does not include storage for the sample schemas.

Storage used is the amount of storage actually used in all data and temporary tablespaces. This does not include storage for the sample schemas. The storage used is the storage in the Autonomous Database as follows:

- Storage used by all database objects. Note: the chart does not include storage for the sample schemas as they do not count against your storage.
- Storage for files users put in the file system.
- Storage used by temporary tablespaces.
- Used storage excludes the free space in the data and temporary tablespaces.

By default the chart does not show the used storage. Select **Storage used** to expand the chart to see used storage (the values are calculated when you open the chart).

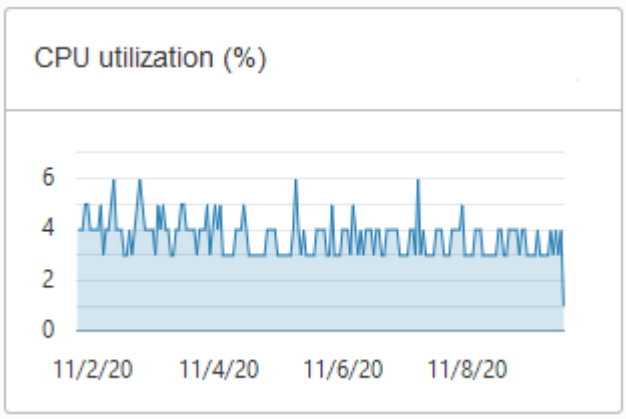
For an Autonomous JSON Database the chart shows an additional field showing the percentage of storage used that is not storing JSON documents.

> **Note:**
>
> If you drop an object, the space continues to be consumed until you empty the recycle bin. See Purging Objects in the Recycle Bin for more information.

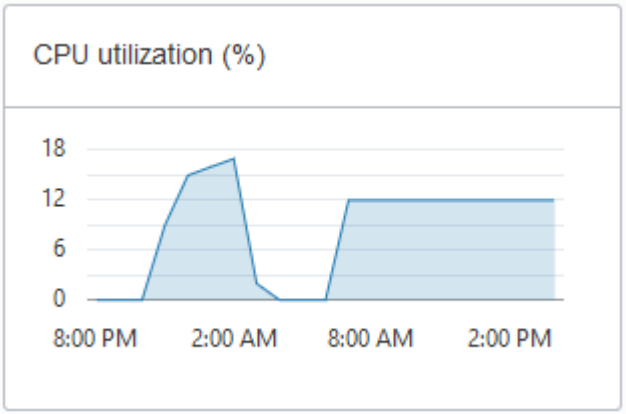
See Use Sample Data Sets in Autonomous Database for information on sample schemas SH and SSB.

- **CPU utilization (%) for ECPU Compute Model:** This chart shows the historical CPU utilization of the service:
  - Compute auto scaling disabled: this chart shows hourly data. A data point shows the average CPU utilization for that hour. For example, a data point at 10:00 shows the average CPU utilization for 9:00-10:00.

    The utilization percentage is reported with respect to the number of ECPUs the database is allowed to use. For example, if the database has four (4) ECPUs, the percentage in this graph is based on 4 ECPUs.

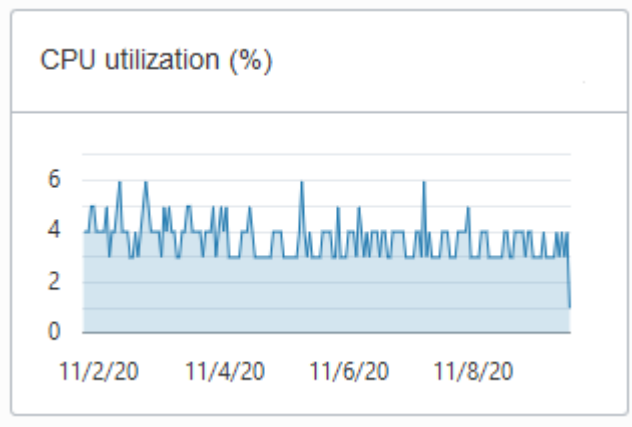


  - Compute auto scaling enabled: For databases with compute auto scaling enabled the utilization percentage is reported with respect to the maximum number of ECPUs the database is allowed to use, which is three times the number of ECPUs. For example, if the database has four (4) ECPUs with auto scaling enabled, the percentage in this graph is based on 12 ECPUs.

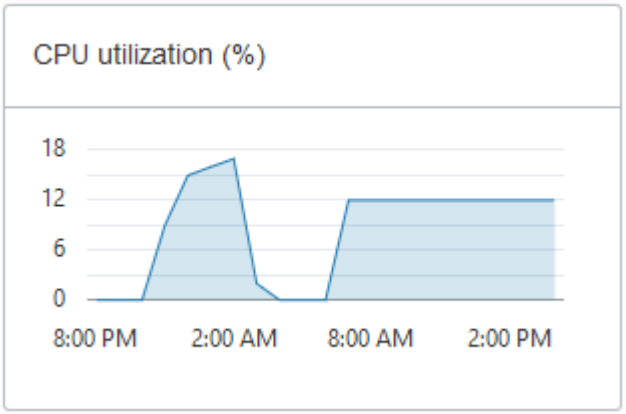


- **CPU utilization (%) for OCPU Compute Model:** This chart shows the historical CPU utilization of the service:
  - Compute auto scaling disabled: this chart shows hourly data. A data point shows the average CPU utilization for that hour. For example, a data point at 10:00 shows the average CPU utilization for 9:00-10:00.

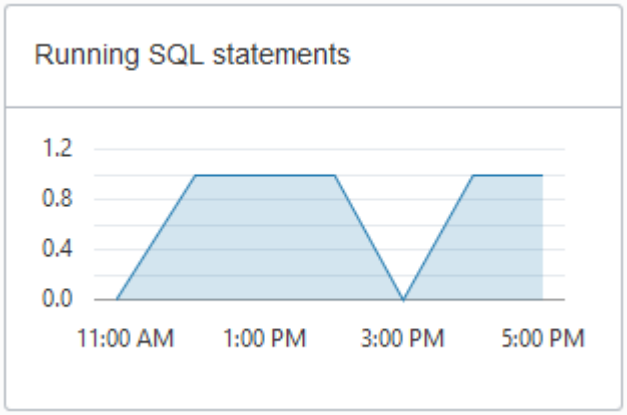
    The utilization percentage is reported with respect to the number of OCPUs the database is allowed to use. For example, if the database has four (4) OCPUs, the percentage in this graph is based on 4 OCPUs.



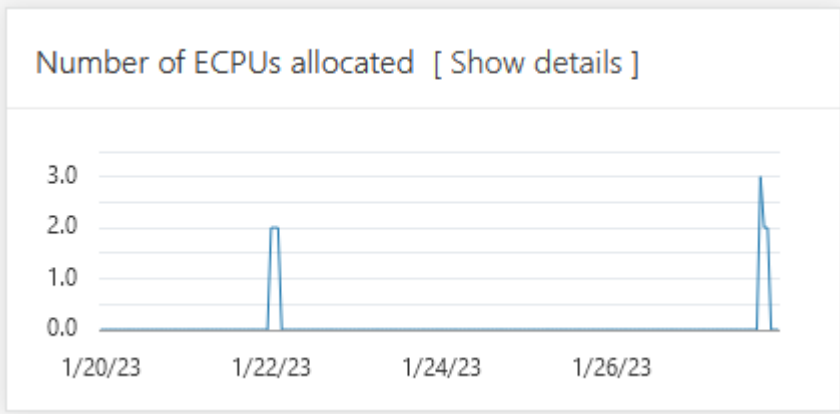
  - Compute auto scaling enabled: For databases with compute auto scaling enabled the utilization percentage is reported with respect to the maximum number of OCPUs the database is allowed to use. For example, if the database has four (4) OCPUs with auto scaling enabled, the percentage in this graph is based on 12 OCPUs.



- **Running SQL statements**: This chart shows the average number of running SQL statements historically. This chart shows hourly data. A data point shows the running SQL statements for that hour. For example, a data point at 10:00 shows the average number of running SQL statements for 9:00-10:00.



- **Number of ECPUs allocated (only shown for ECPU Compute Model):**



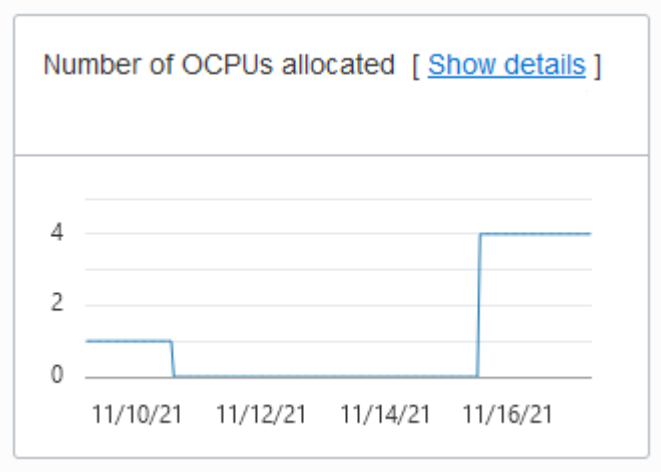
Notes for display results:

- Compute auto scaling disabled: For databases with compute auto scaling disabled, for each hour the chart shows the number of ECPUs allocated to the database if the database is open for at least some part of the hour.
- Compute auto scaling enabled: For databases with compute auto scaling enabled, for each hour the chart shows the average number of ECPUs used during that hour if that value is higher than the number of ECPUs provisioned. If the number of ECPUs used is not higher than the number of ECPUs provisioned, then the chart shows the number of ECPUs allocated for that hour.
- Stopped Database: If the database was stopped for the full hour the chart shows 0 ECPUs allocated for that hour.

Click **Show details** for more information, including the number of ECPUs allocated to the database and to external resources, and the total allocated ECPUs.

The Show details view includes separate values for database ECPU usage and external resource ECPU usage. External resources include: Cloud SQL, Graph, OML4PY, and others. The Total ECPUs are the total number of ECPUs in use on the Autonomous Database. The external ECPUs value shows how external ECPUs contribute to the total ECPU usage.

- **Number of OCPUs allocated (only shown for OCPU Compute Model):**



Notes for display results:

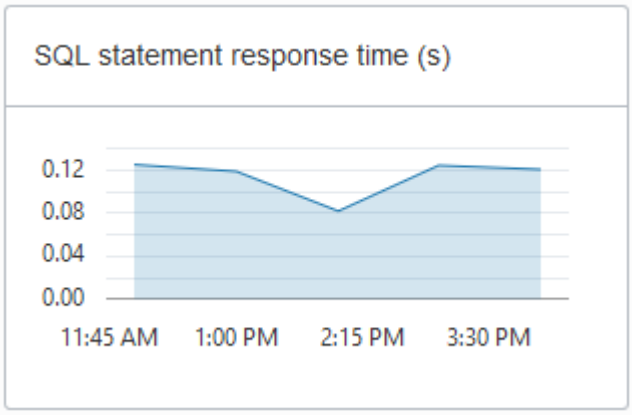
- Compute auto scaling disabled: For databases with compute auto scaling disabled, for each hour the chart shows the number of OCPUs allocated to the database if the database is open for at least some part of the hour.

  - Compute auto scaling enabled: For databases with compute auto scaling enabled, for each hour the chart shows the average number of OCPUs used during that hour if that value is higher than the number of OCPUs provisioned. If the number of OCPUs used is not higher than the number of OCPUs provisioned, then the chart shows the number of OCPUs allocated for that hour.
  - Stopped Database: If the database was stopped for the full hour the chart shows 0 OCPUs allocated for that hour.

  Click **Show details** for more information, including the number of OCPUs allocated to the database and to external resources, and the total allocated OCPUs.

  The Show details view includes separate values for database OCPU usage and external resource OCPU usage. External resources include: Cloud SQL, Graph, OML4PY, and others. The Total OCPUs are the total number of OCPUs in use on the Autonomous Database. The external OCPUs value shows how external OCPUs contribute to the total OCPU usage.

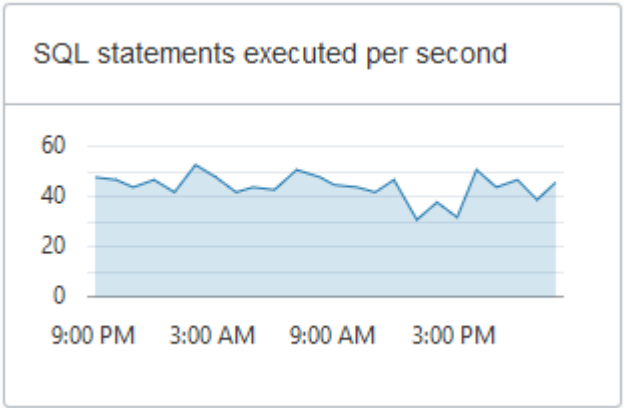
- **SQL statement response time (s)**: This chart shows the average response time, in seconds, of SQL statements historically. This chart shows hourly data. A data point shows the average SQL statement response time for that hour. For example, a data point at 10:00 shows the average SQL statement response time, in seconds, for the hour from 9:00-10:00.



- **SQL statements executed per second**

> **Note:**
>
> Database Dashboard does not show this chart when the Autonomous Database instance workload type is **Data Warehouse**.

.



The default retention period for performance data is thirty (30) days. The CPU utilization, running statements, and average SQL response time charts show data for the last eight (8) days by default.

You can change the retention period by modifying the Automatic Workload Repository retention setting with the PL/SQL procedure `DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS()`. The maximum retention you can set is 30 days. See *Oracle Database PL/SQL Packages and Types Reference*.

If you need to store more performance data you can use the Operations Insights AWR Hub. See Analyze Automatic Workload Repository (AWR) Performance Data for more information.

Database Dashboard Activity

The **Monitor** tab shows real-time and historical information about the Autonomous Database performance data, activity, and utilization.

> **Note:**
>
> The default view in the **Monitor** tab is real-time. This view shows performance data for the last hour.

The charts on this page are:

- **Database Activity**

  This chart shows the average number of sessions in the database using CPU or waiting on a wait event. See *Oracle Database Reference* for more information on wait events.

- **CPU Utilization** (with ECPU compute model)

  This chart shows the CPU utilization of each consumer group. The utilization percentage is reported with respect to the number of ECPUs the database is allowed to use. For example, if the database has four (4) ECPUs, the percentage in this graph is based on 4 ECPUs.

  For databases with compute auto scaling enabled the utilization percentage is reported with respect to the maximum number of ECPUs the database is allowed to use, which is three times the number of ECPUs. For example, if the database has four (4) ECPUs with auto scaling enabled, the percentage in this graph is based on 12 ECPUs.

See Manage Concurrency and Priorities on Autonomous Database for detailed information on consumer groups.

- **CPU Utilization** (with OCPU compute model)

  This chart shows the CPU utilization of each consumer group. The utilization percentage is reported with respect to the number of OCPUs the database is allowed to use. For example, if the database has four (4) OCPUs, the percentage in this graph is based on 4 OCPUs.

  For databases with compute auto scaling enabled the utilization percentage is reported with respect to the maximum number of OCPUs the database is allowed to use. For example, if the database has four (4) OCPUs with auto scaling enabled, the percentage in this graph is based on 12 OCPUs.

  See Manage Concurrency and Priorities on Autonomous Database for detailed information on consumer groups.

- **Running Statements**

  This chart shows the average number of running SQL statements in each consumer group.

  See Manage Concurrency and Priorities on Autonomous Database for detailed information on consumer groups.

- **Queued Statements**

  This chart shows the average number of queued SQL statements in each consumer group.

  See Manage Concurrency and Priorities on Autonomous Database for detailed information on consumer groups.

To see earlier data click **Time period**. The default retention period for performance data is thirty (30) days. By default in the Time Period view the charts show information for the last eight (8) days.

In the time period view you can use the calendar to look at a specific time period in the past 30 days. You can also use the time slider to change the period for which performance data is shown.

> **Note:**
>
> The retention time can be changed by changing the Automatic Workload Repository retention setting with the PL/SQL procedure `DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS`. Be aware that increasing the retention time results in more storage usage for performance data. See *Oracle Database PL/SQL Packages and Types Reference*.

Use Database Actions to Monitor Active Session History Analytics and SQL Statements

In Database Actions the Performance Hub card provides information about Active Session History (ASH) analytics and current and past monitored SQL statements.

Perform the following prerequisite step as necessary:

- Access Database Actions as the ADMIN user. See Access Database Actions a ADMIN for more information.
- The Performance Hub Page

The Performance Hub Page

The Performance Hub page shows performance data for a time period you specify.
To navigate to the Performance Hub page, do either of the following:

- In the Launchpad page, click **Performance Hub**.
- Click **Selector** to display the navigation menu. Under Monitoring, select **Performance Hub**.

> **Note:**
>
> The Performance Hub page is available in the following user interface languages: French, Japanese, Korean, Traditional Chinese, and Simplified Chinese. If you change the language to German, Spanish, Italian, or Portuguese in Preferences, the Performance Hub page reverts to English.

The Performance Hub page consists of these parts:

- **Time Range Area**: Use the controls in time range area at the top of the page to specify the time period for which you want to view performance data.
- **ASH Analytics Tab**: Use this tab to explore ASH (Active Session History) information across a variety of different dimensions for the specified time period.
- **SQL Monitoring Tab**: Use this tab to view the top 100 SQL statement executions by different dimensions for the specified time period, and to view details of SQL statement executions you select.

See Performance Hub in the Database service documentation for more information.

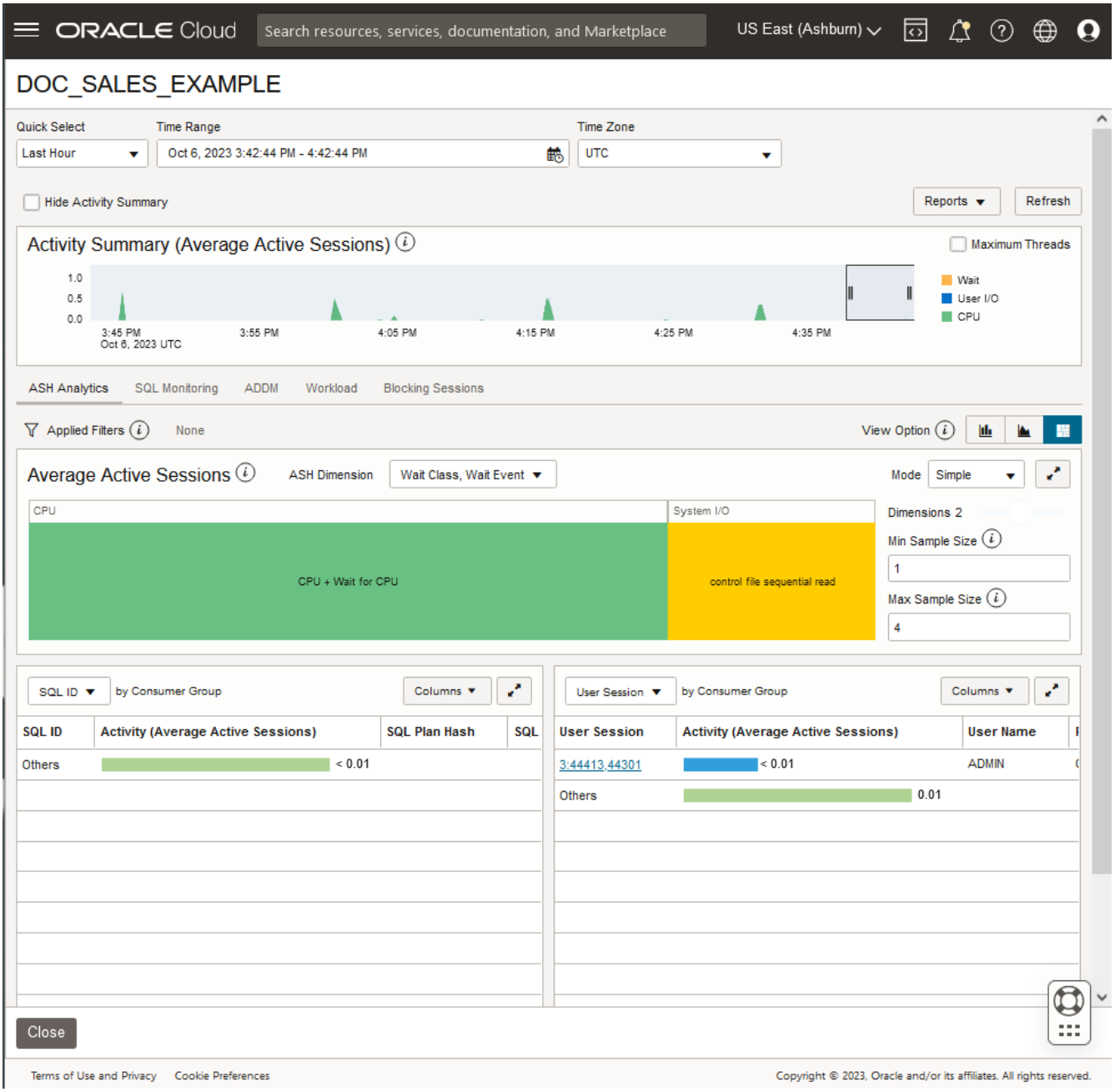
Monitor Autonomous Database with Performance Hub

Use Performance Hub to monitor your database for a defined time period and download statistical reports. Performance Hub also lets you view real-time and historical performance data for an Autonomous Database instance.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database**, and then click Autonomous Database.

1. On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.
2. From the Autonomous Database details page click **Performance Hub**.



See Using Performance Hub to Analyze Database Performance more information.

Component	Description
Top Area	The top of the Performance Hub page shows the following: • **Quick Select**: Use to quickly set the time range to one of Last Hour, Last 8 Hours, Last 24 Hours, or Last Week. • **Time Range**: Use to set the date and time range for Performance Hub to monitor. • **Time Zone**: Select an entry from this list to specify times based on Coordinated Universal Time (UTC), your local web browser time (Browser), or the database time zone setting (Database). • **Activity Summary (Average Active Sessions)**: shows active sessions during the selected time range. It displays the average number of active sessions broken down by CPU, User I/O, and Wait. It also shows the Max CPU usage. You can hide or show the activity summary by selecting **Hide Activity Summary**. • **Reports**: Select to view the Automatic Workload Repository (AWR) report. The AWR report collects, processes, and maintains performance statistics for problem detection and self-tuning purposes. This data is both in memory and stored in the database.
Active Session History (ASH) Analytics	**ASH Analytics** is displayed by default. This tab shows Active Session History (ASH) analytics charts to explore the Active Session History data. You can drill down into database performance across multiple dimensions such as Consumer Group, Wait Class, SQL ID, and User Name. Select an Average Active Sessions dimension and view the top activity for that dimension for the selected time period. For more information, see ASH Analytics.
SQL Monitoring	SQL statements are only monitored if they've been running for at least five seconds or if they're run in parallel. The table displays monitored SQL statement executions by dimensions including Last Active Time, CPU Time, and Database Time. The table displays currently running SQL statements and SQL statements that completed, failed, or were terminated. The columns in the table provide information for monitored SQL statements including Status, Duration, and SQL ID. For more information, see SQL Monitoring.
Automatic Database Diagnostic Monitor (ADDM)	**ADDM** tab provides access to analysis information gathered by the Automatic Database Diagnostic Monitor (ADDM) tool. ADDM analyzes AWR (Automatic Workload Repository) snapshots on a regular basis, locates root causes of any performance problems, provides recommendations for correcting the problems, and identifies non-problem areas of the system. AWR is a repository of historical performance data, therefore ADDM can analyze performance issues after the event, often saving time and resources in reproducing a problem. For more information, see Automatic Database Diagnostic Monitor (ADDM).

Component	Description
Workload	Use the **Workload** tab to visually monitor the database workload to identify spikes and bottlenecks. This tab shows four chart areas that show database workload in various ways: • **CPU Statistics**: Charts CPU usage. • **Wait Time Statistics**: Shows the wait time across the database's foreground sessions, divided by wait classes. • **Workload Profile**: Charts user (client) workload on the database. • **Sessions**: Shows the number of sessions, successful logons, and current logons. For more information, see Workload.
Blocking Sessions	**Blocking Sessions** hierarchically lists sessions that are waiting or are blocked by sessions that are waiting. You can set the minimum wait time required for sessions to be displayed in the list, and you can view a variety of information about a session to determine whether to let it continue or to end it. For more information, see Blocking Sessions.

See Using Performance Hub to Analyze Database Performance more information.

Manage Concurrency and Priorities on Autonomous Database

Concurrency and prioritization of user requests in Autonomous Database is determined by the database service the user is connected with, and whether compute auto scaling is enabled.

- Database Service Names for Autonomous Data Warehouse
  You are required to select a service when you connect to the database. The service names for Autonomous Data Warehouse connections are in the format:
- Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database
  You are required to select a service when you connect to the database. The service names for connecting to Autonomous Transaction Processing or Autonomous JSON Database are in the format:
- Idle Time Limits
  Autonomous Database has predefined idle time limits for sessions so that idle sessions do not hold system resources for a long time.
- Service Concurrency
  The consumer groups of the predefined service names provide different levels of performance and concurrency. The available service names are different depending on your workload: Data Warehouse, Transaction Processing, or JSON Database.
- Change MEDIUM Service Concurrency Limit (ECPU Compute Model)
  If your application requires customized concurrency, you can modify the concurrency limit for your Autonomous Database MEDIUM service.
- Change MEDIUM Service Concurrency Limit (OCPU Compute Model)
  If your application requires customized concurrency, you can modify the concurrency limit for your Autonomous Database MEDIUM service.
- Predefined Job Classes with Oracle Scheduler
  Autonomous Database includes predefined `job_class` values to use with Oracle Scheduler.

Database Service Names for Autonomous Data Warehouse

You are required to select a service when you connect to the database. The service names for Autonomous Data Warehouse connections are in the format:

- *databasename*_high
- *databasename*_medium
- *databasename*_low

These services map to the `LOW`, `MEDIUM`, and `HIGH` consumer groups.

For example, if you create an Autonomous Database with a Data Warehouse workload type and specify the database name as `DB2024`, your service names are:

- db2024_high
- db2024_medium
- db2024_low

If you connect using the `db2024_low` service, the connection uses the `LOW` consumer group.

The basic characteristics of these consumer groups are:

- HIGH: Highest resources, lowest concurrency. Queries run in parallel.
- MEDIUM: Less resources, higher concurrency. Queries run in parallel.

  Picking one of the predefined services provides concurrency values that work well for most applications. In cases where selecting one of the default services does not meet your application's performance needs, you can use the MEDIUM service and modify the concurrency limit. For example, when you run single-user benchmarks, you can set the concurrency limit of the MEDIUM service to 1 in order to obtain the highest degree of parallelism (DOP).

  Depending on your compute model, ECPU or OCPU, see the following for more information.

  – Change MEDIUM Service Concurrency Limit (ECPU Compute Model)
  – Change MEDIUM Service Concurrency Limit (OCPU Compute Model)
- LOW: Least resources, highest concurrency. Queries run serially.

> **Note:**
>
> After connecting to the database using one service, do not attempt to manually switch that connection to a different service by simply changing the consumer group of the connection. When you connect using a service, Autonomous Database performs more actions to configure the connection than just setting its consumer group. You can use the procedure `CS_SESSION.SWITCH_SERVICE` to switch to a different service.
> See SWITCH_SERVICE Procedure for more information.

Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database

You are required to select a service when you connect to the database. The service names for connecting to Autonomous Transaction Processing or Autonomous JSON Database are in the format:

- *databasename*_tpurgent
- *databasename*_tp
- *databasename*_high
- *databasename*_medium
- *databasename*_low

These services map to the `TPURGENT`, `TP`, `HIGH`, `MEDIUM` and `LOW` consumer groups.

For example, if you create an Autonomous Database with a Transaction Processing workload type and specify the database name as `DB2024`, your connection service names are:

- db2024_tpurgent
- db2024_tp
- db2024_high
- db2024_medium
- db2024_low

If you connect using the `db2024_tp` service, the connection uses the `TP` consumer group.

The basic characteristics of these consumer groups are:

- TPURGENT: The highest priority application connection service for time critical transaction processing operations. This connection service supports manual parallelism.
- TP: A typical application connection service for transaction processing operations. This connection service does not run with parallelism.
- HIGH: A high priority application connection service for reporting and batch operations. All operations run in parallel and are subject to queuing.
- MEDIUM: A typical application connection service for reporting and batch operations. All operations run in parallel and are subject to queuing.

  Picking one of the predefined services provides concurrency values that work well for most applications. In cases where selecting one of the default services does not meet your application's performance needs, you can use the MEDIUM service and modify the concurrency limit. For example, when you run single-user benchmarks, you can set the concurrency limit of the MEDIUM service to 1 in order to obtain the highest degree of parallelism (DOP).

  Depending on your compute model, ECPU or OCPU, see the following for more information.

  – Change MEDIUM Service Concurrency Limit (ECPU Compute Model)
  – Change MEDIUM Service Concurrency Limit (OCPU Compute Model)
- LOW: A lowest priority application connection service for reporting or batch processing operations. This connection service does not run with parallelism.

> **Note:**
>
> After connecting to the database using one service, do not attempt to manually switch that connection to a different service by simply changing the consumer group of the connection. When you connect using a service, Autonomous Database performs more actions to configure the connection than just setting its consumer group. You can use the procedure `CS_SESSION.SWITCH_SERVICE` to switch to a different service.
> See SWITCH_SERVICE Procedure for more information.

Idle Time Limits

Autonomous Database has predefined idle time limits for sessions so that idle sessions do not hold system resources for a long time.

A session may be terminated if it stays idle for more than five (5) minutes and the resources it consumes are needed by other users. This allows other active sessions to proceed without waiting for the idle session.

If you want sessions to be terminated after a certain amount of time, independent of the consumed resources needed by other users, then set the `MAX_IDLE_TIME` initialization parameter. The `MAX_IDLE_TIME` parameter specifies the maximum number of minutes that a session can be idle. After the specified amount of time, `MAX_IDLE_TIME` kills sessions.

> **Note:**
>
> Sessions that are idle for more than 48 hours are terminated whether they are holding resources or not.

See MAX_IDLE_TIME for more information.

Service Concurrency

The consumer groups of the predefined service names provide different levels of performance and concurrency. The available service names are different depending on your workload: Data Warehouse, Transaction Processing, or JSON Database.

Picking one of the predefined services provides concurrency values that work well for most applications. In cases where selecting one of the default services does not meet your application's performance needs, you can use the MEDIUM service and modify the concurrency limit. For example, when you run single-user benchmarks, you can set the concurrency limit of the MEDIUM service to 1 in order to obtain the highest degree of parallelism (DOP).

In this topic, the "number of ECPUs" is the **ECPU count** shown in the Oracle Cloud Infrastructure Console. Likewise, if your database uses the OCPU compute model, the "number of OCPUs" is the **OCPU count**.

> **Note:**
>
> OCPU is a legacy billing metric and has been retired for Autonomous Data Warehouse (**Data Warehouse** workload type) and Autonomous Transaction Processing (**Transaction Processing** workload type). Oracle recommends using ECPUs for all new and existing Autonomous Database deployments. See Oracle Support Document 2998742.1 for more information.

- Service Concurrency Limits for Data Warehouse Workloads (ECPU Compute Model)
  The `tnsnames.ora` file provided with the credentials zip file contains three database service names identifiable as high, medium, and low for Autonomous Database with Data Warehouse workloads.
- Service Concurrency Limits for Transaction Processing Workloads (ECPU Compute Model)
  The `tnsnames.ora` file provided with the credentials zip file contains five database service names identifiable as tpurgent, tp, high, medium, and low for Autonomous Database with Transaction Processing or JSON workloads.
- Service Concurrency Limits for Data Warehouse Workloads (OCPU Compute Model)
  The `tnsnames.ora` file provided with the credentials zip file contains three database service names identifiable as high, medium and low for Autonomous Database with Data Warehouse workloads.
- Service Concurrency Limits for Transaction Processing and JSON Database Workloads (OCPU Compute Model)
  The `tnsnames.ora` file provided with the credentials zip file contains five database service names identifiable as tpurgent, tp, high, medium, and low for Autonomous Database with Transaction Processing or with JSON Database workloads.

Service Concurrency Limits for Data Warehouse Workloads (ECPU Compute Model)

The `tnsnames.ora` file provided with the credentials zip file contains three database service names identifiable as high, medium, and low for Autonomous Database with Data Warehouse workloads.

The following shows the details for the number of concurrent statements for each connection service for Data Warehouse workloads, with **Compute auto scaling** disabled, and with **Compute auto scaling** enabled.

> **Note:**
>
> The values in this table apply when the number of ECPUs is equal to or greater than 4.
> When the number of ECPUs is 2, all services use the concurrency limit 150. When the number of ECPUs is 3, all services use the concurrency limit 225 (this applies for **Compute auto scaling** enabled or disabled).

Database Service Name	Number of Concurrent Queries with Compute Auto Scaling Disabled	Number of Concurrent Queries with Compute Auto Scaling Enabled
`high`	3	9

Database Service Name	Number of Concurrent Queries with Compute Auto Scaling Disabled	Number of Concurrent Queries with Compute Auto Scaling Enabled
`medium`	0.25125 × number of ECPUs A decimal result is truncated.	0.75375 × number of ECPUs A decimal result is truncated.
`low`	75 × number of ECPUs	75 × number of ECPUs

When these concurrency levels are reached for the MEDIUM and HIGH consumer groups, new SQL statements in that consumer group will be queued until one or more running statements finish. With the LOW consumer group, when the concurrency limit is reached you will not be able to connect new sessions.

For example, with auto scaling disabled on an Autonomous Database instance with 32 ECPUs, the HIGH consumer group will be able to run 3 concurrent SQL statements when the MEDIUM consumer group is not running any statements. The MEDIUM consumer group will be able to run 8 concurrent SQL statements when the HIGH consumer group is not running any statements. The LOW consumer group will be able to run 2400 concurrent SQL statements.

> **Note:**
>
> The HIGH consumer group can run at least 1 SQL statement when the MEDIUM consumer group is also running statements.

For this example with auto scaling enabled on an Autonomous Database instance with 32 ECPUs, the HIGH consumer group will be able to run 9 concurrent SQL statements when the MEDIUM consumer group is not running any statements. The MEDIUM consumer group will be able to run 24 concurrent SQL statements when the HIGH consumer group is not running any statements. The LOW consumer group will be able to run 2400 concurrent SQL statements.

The following table shows sample concurrent connections values for a database with 32 ECPUs with **Compute auto scaling** disabled and with **Compute auto scaling** enabled.

Database Service Name	Number of Concurrent Queries with Compute Auto Scaling Disabled	Number of Concurrent Queries with Compute Auto Scaling Enabled
`high`	3	9
`medium`	8	24
`low`	Up to 2400	Up to 2400

See Change MEDIUM Service Concurrency Limit (ECPU Compute Model) for more information.

Service Concurrency Limits for Transaction Processing Workloads (ECPU Compute Model)

The `tnsnames.ora` file provided with the credentials zip file contains five database service names identifiable as tpurgent, tp, high, medium, and low for Autonomous Database with Transaction Processing or JSON workloads.

The following shows the details for the default number of concurrent statements for each connection service for Transaction Processing or JSON workloads.

> **Note:**
>
> The values in this table apply when the number of ECPUs is equal to or greater than 4.
> When the number of ECPUs is 2, all services use the concurrency limit 150. When the number of ECPUs is 3, all services use the concurrency limit 225 (this applies for **Compute auto scaling** enabled or disabled).

Database Service Name	Concurrent Statements with Compute Auto Scaling Disabled	Concurrent Statements with Compute Auto Scaling Enabled
`tpurgent`	75 × number of ECPUs	75 × number of ECPUs
`tp`	75 × number of ECPUs	75 × number of ECPUs
`high`	3	9
`medium`	0.25125 × number of ECPUs A decimal result is truncated.	0.75375 × number of ECPUs A decimal result is truncated.
`low`	75 × number of ECPUs	75 × number of ECPUs

See Change MEDIUM Service Concurrency Limit (ECPU Compute Model) for more information.

Service Concurrency Limits for Data Warehouse Workloads (OCPU Compute Model)

The `tnsnames.ora` file provided with the credentials zip file contains three database service names identifiable as high, medium and low for Autonomous Database with Data Warehouse workloads.

The following shows the details for the number of concurrent statements for each connection service for Data Warehouse workloads, with OCPU auto scaling disabled, and with OCPU auto scaling enabled.

> **Note:**
>
> The values in this table apply when the number of OCPUs is greater than 1. With 1 OCPU, the concurrency limit for each service is 300 and the DOP is 1.

Database Service Name	Concurrent Statements with OCPU Auto Scaling Disabled	Concurrent Statements with OCPU Auto Scaling Enabled
`high`	3	9
`medium`	1.26 × number of OCPUs	3.78 × number of OCPUs
`low`	300 × number of OCPUs	300 × number of OCPUs

When these concurrency levels are reached for the MEDIUM and HIGH consumer groups, new SQL statements in that consumer group will be queued until one or more running statements finish. With the LOW consumer group, when the concurrency limit is reached you will not be able to connect new sessions.

For example, for an Autonomous Database instance with 16 OCPUs and OCPU auto scaling disabled, the HIGH consumer group will be able to run 3 concurrent SQL statements when the MEDIUM consumer group is not running any statements. The MEDIUM consumer group will be able to run 20 concurrent SQL statements when the HIGH consumer group is not running any statements. The LOW consumer group will be able to run 4800 concurrent SQL statements. The HIGH consumer group can run at least 1 SQL statement when the MEDIUM consumer group is also running statements. When OCPU auto scaling is enabled, for the HIGH and MEDIUM consumer groups the concurrency values will be three times larger.

The following table shows sample concurrent connections values for a database with 16 OCPUs, showing the values with both OCPU auto scaling enabled and disabled.

Database Service Name	Number of Concurrent Queries with OCPU Auto Scaling Disabled	Number of Concurrent Queries with OCPU Auto Scaling Enabled
`high`	3	9
`medium`	20	60
`low`	Up to 4800	Up to 4800

Service Concurrency Limits for Transaction Processing and JSON Database Workloads (OCPU Compute Model)

The `tnsnames.ora` file provided with the credentials zip file contains five database service names identifiable as tpurgent, tp, high, medium, and low for Autonomous Database with Transaction Processing or with JSON Database workloads.

The following shows the details for the default number of concurrent statements for each connection service for Transaction Processing or JSON Database workloads, with OCPU auto scaling disabled, and with OCPU auto scaling enabled.

> **Note:**
>
> The values in this table apply when the number of OCPUs is greater than 1. With 1 OCPU, the concurrency limit for each service is 300 and the DOP is 1.

Database Service Name	Concurrent Statements with OCPU Auto Scaling Disabled	Concurrent Statements with OCPU Auto Scaling Enabled
`tpurgent`	300 × number of OCPUs	300 × number of OCPUs
`tp`	300 × number of OCPUs	300 × number of OCPUs
`high`	3	9
`medium`	1.26 × number of OCPUs	3.78 × number of OCPUs
`low`	300 × number of OCPUs	300 × number of OCPUs

Change MEDIUM Service Concurrency Limit (ECPU Compute Model)

If your application requires customized concurrency, you can modify the concurrency limit for your Autonomous Database MEDIUM service.

Picking one of the predefined services provides concurrency values that work well for most applications. In cases where selecting one of the default services does not meet your

application's performance needs, you can use the MEDIUM service and modify the concurrency limit. For example, when you run single-user benchmarks, you can set the concurrency limit of the MEDIUM service to 1 in order to obtain the highest degree of parallelism (DOP).

> **Note:**
>
> Changing the concurrency limit is only allowed for an instance that has four (4) or more ECPUs.

For example, with **Compute auto scaling** disabled, if your instance is configured with 400 ECPUs, by default Autonomous Database provides a concurrency limit of 100 for the MEDIUM service:

`0.25125 x number of ECPUs` sessions (up to 100 concurrent queries). A decimal result is truncated.

In this example the MEDIUM service supports an application with up to 100 concurrent queries with DOP of 4. If you only need 50 concurrent queries and you want a higher DOP you can decrease the concurrency limit and the database increases the DOP. To do this, set the MEDIUM service concurrency limit to 50. When you change the concurrency limit the database calculates and sets the DOP based on the concurrency limit you select and the number of ECPUs. For this example, with the concurrency limit set to 50, the new DOP is 12.

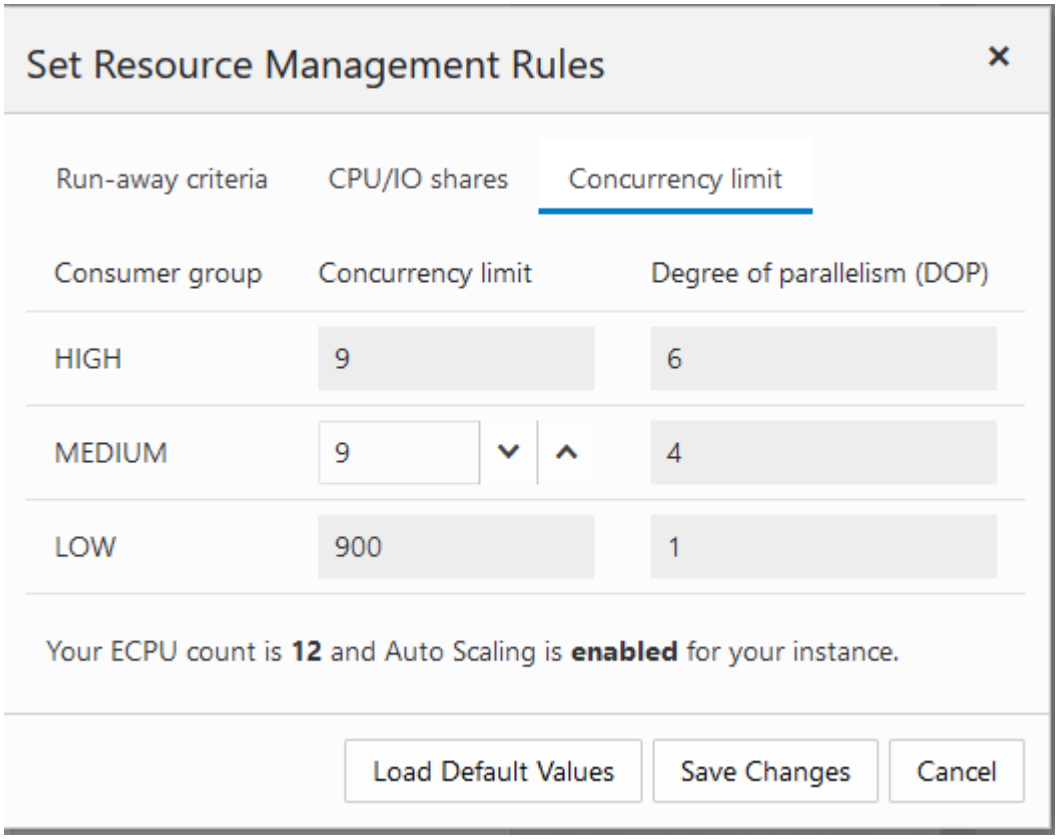
With **Compute auto scaling** enabled, the DOP is set to a value three times greater. In this example the DOP value would be 36.

You can change the concurrency limit for the MEDIUM service in Database Actions or using the PL/SQL package `CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE`.

Follow these steps to change the MEDIUM service concurrency limit in Database Actions:

1. Access Database Actions as the ADMIN user.

   See Access Database Actions as ADMIN for more information.
2. On the Database Actions Launchpad, under **Administration**, click **Set Resource Management Rules**.
3. On the Set Resource Management Rules page, select the **Concurrency limit** tab.



4. For the MEDIUM service, change the value to the desired concurrency limit by entering a value or by clicking the Decrement or Increment icons.

   If the concurrency limit you specify is not valid, based on the number of ECPUs, you will receive a message such as the following, listing the valid range of values for your instance:

   ```
   Enter a number between 1 and 12.
   ```

5. Click **Save Changes**.
6. Click **OK**.

To reset the concurrency limit for the MEDIUM service to its default value, click **Load Default Values** and click **Save Changes**.

- Change MEDIUM Service Concurrency Limit with PL/SQL Procedure UPDATE_PLAN_DIRECTIVE (ECPU Compute Model)
  As an alternative to using the **Set Resource Management Rules** card in Database Actions, you can use PL/SQL to change the concurrency limit for the MEDIUM service.
- Change MEDIUM Service Concurrency Limit Notes (ECPU Compute Model)

Related Topics

- Change MEDIUM Service Concurrency Limit (OCPU Compute Model)
  If your application requires customized concurrency, you can modify the concurrency limit for your Autonomous Database MEDIUM service.

Change MEDIUM Service Concurrency Limit with PL/SQL Procedure UPDATE_PLAN_DIRECTIVE (ECPU Compute Model)

As an alternative to using the **Set Resource Management Rules** card in Database Actions, you can use PL/SQL to change the concurrency limit for the MEDIUM service.

To change the MEDIUM service concurrency limit with `CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE`:

1. Call the PL/SQL procedure `CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE` to update the concurrency limit for the MEDIUM consumer group.

   For example, with 12 ECPUs, change the MEDIUM service's concurrency limit to 2, as follows:

   ```
   BEGIN
       CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'MEDIUM',
   concurrency_limit => 2);
   END;
   /
   ```

   If the `concurrency_limit` you specify is not valid, based on the number of ECPUs, the procedure shows an error message, listing the valid range of values for your instance. For example with 12 ECPUs:

   ```
   ORA-20000: Invalid or missing value. Concurrency limit must be between 1
   and 9 for the specified CPU count
   ```

   This error message example is from an instance with 12 ECPUs.

2. Use the PL/SQL function `CS_RESOURCE_MANAGER.LIST_CURRENT_RULES` to verify the updated MEDIUM service concurrency limit and degree of parallelism:

   ```
   SELECT * FROM CS_RESOURCE_MANAGER.LIST_CURRENT_RULES();
   ```

   This procedure returns the list of values for all consumer groups. After you modify the concurrency limit as specified in Step 1, check the MEDIUM service `CONCURRENCY_LIMIT` and `DEGREE_OF_PARALLELISM` values to verify your changes.

3. After you change the concurrency limit for the MEDIUM service, test your application by connecting with the MEDIUM service to verify that the customized concurrency limit meets your performance objectives.

When you want to go back to the default values, use the `CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES` PL/SQL procedure to revert to the default settings for the MEDIUM service.

For example:

```
BEGIN
    CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'MEDIUM',
concurrency_limit => TRUE);
END;
/
```

See CS_RESOURCE_MANAGER Package for more information.

Change MEDIUM Service Concurrency Limit Notes (ECPU Compute Model)

- Changing the concurrency limit is only allowed for the MEDIUM service.
- Changing the concurrency limit is only allowed when the number of ECPUs is greater than or equal to 4.
- Changing the concurrency limit also changes the degree of parallelism (in some cases the value does not change, depending on the magnitude of the difference between the old concurrency limit and the new value you set).
- The concurrency limit you set must be in the range:
  - With **Compute auto scaling** disabled: between 1 and `.75 x the number of ECPUs`
  - With **Compute auto scaling** enabled: between 1 and `2.25 x the number of ECPUs`
- The MEDIUM service sets the following concurrency limit and DOP values by default:

MEDIUM Database Service	Default Value with Compute Auto Scaling Disabled	Default Value with Compute Auto Scaling Enabled
Concurrency Limit	`0.25125 × number of ECPUs` when the number of ECPUs ≥8 A decimal result is truncated 2 when the number of ECPUs is in the range `4 ≤ ECPUs < 8`	`0.75375 × number of ECPUs` when the number of ECPUs ≥8 A decimal result is truncated 6 when the number of ECPUs is in the range: `4 ≤ ECPUs < 8`
DOP	4 when the number of ECPUs ≥8 or `TRUNC (ECPU/2)`, when the number of ECPUs < 8	4 when the number of ECPUs ≥8 or `TRUNC (ECPU/2)`, when the number of ECPUs < 8

- By changing the value of the concurrency limit, the DOP of the MEDIUM service can go as low as 2 and as high as `.75 * number of ECPUs` (if **Compute auto scaling** is disabled) or `2.25 x number of ECPUs` (if **Compute auto scaling** is enabled).

  See Use Auto Scaling for information on **Compute auto scaling**.
- At any time you can return to the default values for the MEDIUM service concurrency limit and DOP.

Change MEDIUM Service Concurrency Limit (OCPU Compute Model)

If your application requires customized concurrency, you can modify the concurrency limit for your Autonomous Database MEDIUM service.

Picking one of the predefined services provides concurrency values that work well for most applications. In cases where selecting one of the default services does not meet your application's performance needs, you can use the MEDIUM service and modify the concurrency limit. For example, when you run single-user benchmarks, you can set the concurrency limit of the MEDIUM service to 1 in order to obtain the highest degree of parallelism (DOP).

> **Note:**
>
> Changing the concurrency limit is only allowed for an instance that has two (2) or more OCPUs.

For example, if your instance is configured with 100 OCPUs, by default Autonomous Database provides a concurrency limit of 126 for the MEDIUM service:

`1.26 x number of OCPUs` sessions (up to 126 concurrent queries)

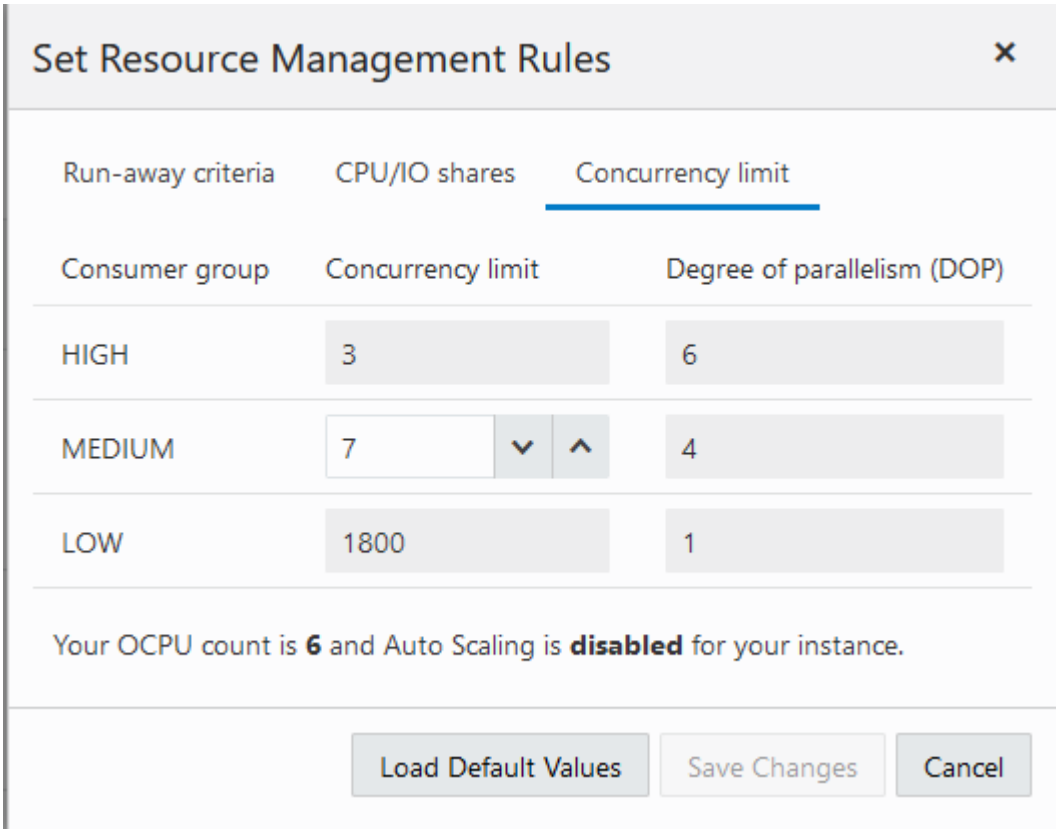
In this example using the MEDIUM service supports an application with up to 126 concurrent queries with DOP of 4. If you only need 50 concurrent queries and you want a higher DOP you can decrease the concurrency limit and thus increase the DOP. To do this, set the MEDIUM service concurrency limit to 50. When you change the concurrency limit the system calculates and sets the DOP based on the concurrency limit you select and the number of OCPUs. For this example, with the concurrency limit set to 50, the new DOP is 12. When **OCPU auto scaling** is enabled, the DOP is set to a value three times greater. In this example the DOP value would be 36.

You can change the concurrency limit for the MEDIUM service in Database Actions or using the PL/SQL package `CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE`.

Follow these steps to change the MEDIUM service concurrency limit in Database Actions:

1. Access Database Actions as the ADMIN user.

   See Access Database Actions as ADMIN for more information.
2. On the Database Actions Launchpad, under **Administration**, click **Set Resource Management Rules**.
3. On the Set Resource Management Rules page, select the **Concurrency limit** tab.



4. For the MEDIUM service, change the value to the desired concurrency limit by entering a value or by clicking the Decrement or Increment icons.

   If the concurrency limit you specify is not valid, based on the number of OCPUs, you will receive a message such as the following, listing the valid range of values for your instance:

   ```
   Please enter a concurrency limit between 1 and 300
   ```

   This error message example is from an instance with 100 OCPUs (the 300 maximum value shown is 3 x number of OCPUs).

5. Click **Save Changes**.
6. Click **OK**.

To reset the concurrency limit for the MEDIUM service to its default value, click **Load Default Values** and click **Save Changes**.

- Change MEDIUM Service Concurrency Limit with PL/SQL Procedure UPDATE_PLAN_DIRECTIVE (OCPU Compute Model)
  As an alternative to using the **Set Resource Management Rules** card in Database Actions, you can use PL/SQL to change the concurrency limit for the MEDIUM service.
- Change MEDIUM Service Concurrency Limit Notes (OCPU Compute Model)

Related Topics

- Change MEDIUM Service Concurrency Limit (ECPU Compute Model)
  If your application requires customized concurrency, you can modify the concurrency limit for your Autonomous Database MEDIUM service.

Change MEDIUM Service Concurrency Limit with PL/SQL Procedure UPDATE_PLAN_DIRECTIVE (OCPU Compute Model)

As an alternative to using the **Set Resource Management Rules** card in Database Actions, you can use PL/SQL to change the concurrency limit for the MEDIUM service.

To change the MEDIUM service concurrency limit with `CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE`:

1. Call the PL/SQL procedure `CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE` to update the concurrency limit for the MEDIUM consumer group.

   For example, with 3 OCPUs, change the MEDIUM service's concurrency limit to 2, as follows:

   ```
   BEGIN
       CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'MEDIUM',
   concurrency_limit => 2);
   END;
   /
   ```

   If the `concurrency_limit` you specify is not valid, based on the number of OCPUs, you will receive a message such as the following, listing the valid range of values for your instance:

   ```
   ORA-20000: Invalid or missing value. Concurrency limit must be between 1
   and 9 for the specified OCPU count
   ```

   This error message example is from an instance with 3 OCPUs.

2. Use the PL/SQL function `CS_RESOURCE_MANAGER.LIST_CURRENT_RULES` to verify the updated MEDIUM service concurrency limit and degree of parallelism:

   ```
   SELECT * FROM CS_RESOURCE_MANAGER.LIST_CURRENT_RULES();

   CONSUMER_GROUP ELAPSED_TIME_LIMIT IO_MEGABYTES_LIMIT SHARES
   CONCURRENCY_LIMIT DEGREE_OF_PARALLELISM
   -------------- ------------------ ------------------ ------
   ----------------- ---------------------
   HIGH
   4                 3                      3
   MEDIUM
   2                 2                      9
   LOW                                                        1
   900                     1
   ```

   This procedure returns the list of values for all consumer groups. After you modify the concurrency limit as specified in Step 1, check the MEDIUM service `CONCURRENCY_LIMIT` and `DEGREE_OF_PARALLELISM` values to verify your changes.

3. After you change the concurrency limit for the MEDIUM service, test your application by connecting with the MEDIUM service to verify that the customized concurrency limit meets your performance objectives.

When you want to go back to the default values, use the `CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES` PL/SQL procedure to revert to the default settings for the MEDIUM service.

For example:

```
BEGIN
    CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'MEDIUM',
concurrency_limit => TRUE);
END;
/
```

See CS_RESOURCE_MANAGER Package for more information.

Change MEDIUM Service Concurrency Limit Notes (OCPU Compute Model)

- Changing the concurrency limit is only allowed for the MEDIUM service.
- Changing the concurrency limit is only allowed when the number of OCPUs is greater than 1.
- Changing the concurrency limit also changes the degree of parallelism (in most cases, depending on the magnitude of the difference between the old concurrency limit and the new value you set).
- The concurrency limit you set must be in the range:
  - With **OCPU auto scaling** disabled: between: `1` and `3 x the number of OCPUs`
  - With **OCPU auto scaling** enabled: between `1` and `9 x the number of OCPUs`
- The MEDIUM service sets the following concurrency limit and DOP values by default:

MEDIUM Database Service	Default Value with OCPU Auto Scaling Disabled	Default Value with OCPU Auto Scaling Enabled
Concurrency Limit	1.26 × number of OCPUs when the number of OCPUs ≥4 5 when the number of OCPUs < 4	3.78 × number of OCPUs when the number of OCPUs ≥4 15 when the number of OCPUs < 4
DOP	4 when the number of OCPUs ≥4 or The number of OCPUs, when the number of OCPUs < 4	4 when the number of OCPUs ≥4 or The number of OCPUs, when the number of OCPUs < 4

- By changing the value of the concurrency limit, the DOP of the MEDIUM service can go as low as 2 and as high as: `2 x number of OCPUs` (if compute auto scaling is disabled) or `6 x number of OCPUs` (if compute auto scaling is enabled).

  See Use Auto Scaling for information on compute auto scaling.
- At any time you can return to the default values for the MEDIUM service concurrency limit and DOP.

Predefined Job Classes with Oracle Scheduler

Autonomous Database includes predefined `job_class` values to use with Oracle Scheduler.

The predefined `job_class` values, `TPURGENT`, `TP`, `HIGH`, `MEDIUM` and `LOW` map to the corresponding consumer groups. These job classes allow you to specify the consumer group a job runs in with `DBMS_SCHEDULER.CREATE_JOB`.

The `DBMS_SCHEDULER.CREATE_JOB` procedure supports `PLSQL_BLOCK` and `STORED_PROCEDURE` job types for the `job_type` parameter in Autonomous Database.

For example: use the following to create a single regular job to run in `HIGH` consumer group:

```
BEGIN
   DBMS_SCHEDULER.CREATE_JOB (
     job_name => 'update_sales',
     job_type => 'STORED_PROCEDURE',
     job_action => 'OPS.SALES_PKG.UPDATE_SALES_SUMMARY',
     start_date => '28-APR-19 07.00.00 PM Australia/Sydney',
     repeat_interval => 'FREQ=DAILY;INTERVAL=2',
     end_date => '20-NOV-19 07.00.00 PM Australia/Sydney',
     auto_drop => FALSE,
     job_class => 'HIGH',
     comments => 'My new job');
END;
/
```

Notes for Oracle Scheduler:

- To use `DBMS_SCHEDULER.CREATE_JOB` additional grants for specific roles or privileges might be required. The `ADMIN` user and users with `DWROLE` have the required `CREATE SESSION` and `CREATE JOB` privileges. If a user does not have `DWROLE` then grants are required for `CREATE SESSION` and `CREATE JOB` privileges.
- The `instance_id` job attribute is ignored for Oracle Scheduler jobs running on Autonomous Database.

See Scheduling Jobs with Oracle Scheduler for more information on Oracle Scheduler and `DBMS_SCHEDULER.CREATE_JOB`.

See SET_ATTRIBUTE Procedure for information on job attributes.

Manage CPU/IO Shares on Autonomous Database

Autonomous Database comes with predefined CPU/IO shares assigned to different consumer groups. You can modify these predefined CPU/IO shares if your workload requires different CPU/IO resource allocations.

The CPU/IO shares assigned to the consumer groups determine the CPU/IO resources a consumer group can use with respect to the other consumer groups. The default CPU/IO shares depend on the Autonomous Database workload.

Workload Type	Details
Data Warehouse	By default, the CPU/IO shares assigned to the consumer groups HIGH, MEDIUM, LOW are 4, 2, and 1, respectively. With the default settings the consumer group HIGH will be able to use 4 times more CPU/IO resources compared to LOW and 2 times more CPU/IO resources compared to MEDIUM, when needed. The consumer group MEDIUM will be able to use 2 times more CPU/IO resources compared to LOW, when needed.

Workload Type	Details
Transaction Processing JSON Database	By default, the CPU/IO shares assigned to the consumer groups TPURGENT, TP, HIGH, MEDIUM, and LOW are 12, 8, 4, 2, and 1, respectively. With the default settings the consumer group TPURGENT will be able to use 12 times more CPU/IO resources compared to LOW, when needed. The consumer group TP will be able to use 4 times more CPU/IO resources compared to MEDIUM, when needed.

You can set CPU/IO shares in Database Actions or using the PL/SQL package `CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE`.

To use Database Actions to change the CPU/IO share values for consumer groups:

1. Access Database Actions as the ADMIN user.

   See Access Database Actions as ADMIN for more information.
2. On the Database Actions Launchpad, under **Administration**, click **Set Resource Management Rules**.
3. Select the **CPU/IO shares** tab to set CPU/IO share values for consumer groups.
4. Set the desired CPU/IO share value for a consumer group by entering a value or by clicking the Decrement or Increment icons.
5. Click **Save Changes**.
6. Click **OK**.

To reset CPU/IO shares values to the defaults, click **Load Default Values** and click **Save Changes** to apply the populated values.

As an alternative to using Database Actions, you can use the PL/SQL procedure `CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE` to change the CPU/IO share values for consumer groups.

For example, on an Autonomous Data Warehouse database, run the following script as the `ADMIN` user to set the CPU/IO shares to 8, 2, and 1 for consumer groups HIGH, MEDIUM, and LOW respectively. This allows the consumer group HIGH to use 4 times more CPU/IO resources compared to the consumer group MEDIUM and 8 times CPU/IO resources compared to the consumer group LOW:

```
BEGIN
   CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'HIGH', shares => 8);
   CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'MEDIUM', shares => 2);
   CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'LOW', shares => 1);
END;
/
```

For example, on an Autonomous JSON Database or on an Autonomous Transaction Processing database, run the following script as the ADMIN user to set CPU/IO shares to 12, 4, 2, 1, and 1 for the consumer groups TPURGENT, TP, HIGH, MEDIUM, and LOW respectively. This allows the consumer group TPURGENT to use 3 times more CPU/IO resources compared to the consumer group TP and 12 times CPU/IO resources compared to the consumer group MEDIUM:

```
BEGIN
   CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'TPURGENT', shares => 12);
```

```
   CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'TP', shares => 4);
   CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'HIGH', shares => 2);
   CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'MEDIUM', shares => 1);
   CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'LOW', shares => 1);
END;
/
```

When you want to go back to the default shares values you can use the PL/SQL procedure `CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES` to revert to the default settings.

For example, on an Autonomous Data Warehouse database, run the following script as the `ADMIN` user to set the CPU/IO shares to default values for the consumer groups HIGH, MEDIUM, and LOW:

```
BEGIN
   CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'HIGH', shares => TRUE);
   CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'MEDIUM', shares => TRUE);
   CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'LOW', shares => TRUE);
END;
/
```

For example, on an or on an Autonomous JSON Database or on an Autonomous Transaction Processing database, run the following script as the ADMIN user to set the default values for CPU/IO shares for the consumer groups TPURGENT, TP, HIGH, MEDIUM, and LOW:

```
BEGIN
   CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'TPURGENT', shares => TRUE);
   CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'TP', shares => TRUE);
   CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'HIGH', shares => TRUE);
   CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'MEDIUM', shares => TRUE);
   CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'LOW', shares => TRUE);
END;
/
```

See CS_RESOURCE_MANAGER Package for more information.

Manage Runaway SQL Statements on Autonomous Database

Specifies how you configure Autonomous Database to terminate SQL statements automatically based on their query runtime or their IO usage.

You can set runtime run-away rules for query run time and IO usage in Database Actions or using the PL/SQL package `CS_RESOURCE_MANAGER`.

Follow these steps to use Database Actions to set runtime usage rules:

1. Access Database Actions as the ADMIN user.

   See Access Database Actions as ADMIN for more information.
2. On the Database Actions Launchpad, under **Administration**, click **Set Resource Management Rules**.
3. Select the **Run-away criteria** tab to set usage rules for a consumer group.
4. Select the **Consumer group**.

5. Set runaway criteria values:
   - **Query run time (seconds)**
   - **Amount of IO (MB)**
6. Click **Save Changes**.
7. Click **OK**.

When a SQL statement in the specified consumer group runs more than the specified runtime limit or does more IO than the specified amount, then the SQL statement will be terminated.

Click **Load Default Values** to load the default values; then click **Save Changes** to apply the populated values.

You can also use the procedure `CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE` to set these rules. For example, to set a runtime limit of 120 seconds and an IO limit of 1000MB for the HIGH consumer group run the following command when connected to the database as the ADMIN user:

```
BEGIN
     CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'HIGH',
io_megabytes_limit => 1000, elapsed_time_limit => 120);
END;
/
```

To reset the values and lift the limits, you can set the values to null:

```
BEGIN
     CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'HIGH',
io_megabytes_limit => null, elapsed_time_limit => null);
END;
/
```

See CS_RESOURCE_MANAGER Package for more information.

Use Fast Ingest on Autonomous Database

Fast ingest optimizes the processing of high-frequency, single-row data inserts into the database from applications, such as Internet of Things (IoT) applications, to improve data insert performance.

The intent of fast-ingest is to support applications that generate lots of informational data that has important value in the aggregate but that doesn't necessarily require full ACID guarantees. Many applications in the Internet of Things (IoT) have a rapid "fire and forget" type workload, such as sensor data, smart meter data or even traffic cameras. For these applications, data might be collected and written to the database in high volumes for later analysis.

Fast ingest is very different from normal Oracle Database transaction processing where data is logged and never lost once "written" to the database (that is, committed). In order to achieve the maximum ingest throughput, the normal Oracle transaction mechanisms are bypassed, and it is the responsibility of the application to check to see that all data was indeed written to the database. Special APIs have been added that can be called to check if the data has been written to the database.

For information on fast ingest and the steps involved in using this feature, refer to Using Fast Ingest in *Database Performance Tuning Guide*.

In addition, the following are required to use Fast Ingest on Autonomous Database:

- **Enable the Optimizer to Use Hints**:
  To use fast ingest with Autonomous Database, you must enable the optimizer to use hints by setting the `optimizer_ignore_hints` parameter to `FALSE` at the session or system level, as appropriate.

  Depending on your Autonomous Database workload type, by default `optimizer_ignore_hints` may be set to `FALSE` at the system level. See Manage Optimizer Statistics on Autonomous Database for more information.
- **Create a Table for Fast Ingest**:
  Prerequisites for Fast Ingest Table contains the limitations for tables to be eligible for Fast Ingest (tables with the specified characteristics cannot use fast ingest).

Monitor the Performance of Autonomous Database with Oracle Management Cloud

Oracle Management Cloud allows you to monitor availability and performance for Autonomous Databases. You can use Oracle Database Management, part of Oracle Management Cloud, to monitor Autonomous Databases and On-premise Oracle Databases.

For information on using Oracle Management Cloud with Autonomous Database see the following:

- *Using Oracle Database Management for Autonomous Databases*
- *Getting Started with Oracle Management Cloud*

Monitor Performance with Autonomous Database Metrics

You can monitor the health, capacity, and performance of your databases with metrics, alarms, and notifications. You can use Oracle Cloud Infrastructure Console or Monitoring APIs to view metrics.

- View Metrics for an Autonomous Database Instance
  Shows the steps to view the Autonomous Database metrics.
- View Metrics for Autonomous Databases in a Compartment
  Shows the steps to view metrics for Autonomous Databases in a compartment.
- Autonomous Database Metrics and Dimensions
  You can limit the instances where you see metrics with dimensions. The available dimensions include: workload type, instance display name, region, and the instance OCID.

View Metrics for an Autonomous Database Instance

Shows the steps to view the Autonomous Database metrics.

> **Note:**
>
> To view metrics you must have the required access as specified in an Oracle Cloud Infrastructure policy (whether you're using the Console, the REST API, or another tool). See Getting Started with Policies for information on policies.

Perform the following steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To view metrics for an Autonomous Database instance:

1. On the Details page, under **Resources**, click **Metrics**.
2. There is a chart for each metric. In each chart you can select the **Interval** and **Statistic** or use the default values.

The following list shows the default Oracle Cloud Infrastructure Console metrics. See Available Metrics: oci_autonomous_database for a list of the database metrics and dimensions.

Metric Name	Description
CPU Utilization	CPU utilization expressed as a percentage, aggregated across all consumer groups. The utilization percentage is reported with respect to the number of CPUs the database is allowed to use, which is the number of ECPUs. If your database uses OCPUs, the number of CPUs allowed is two times the number of OCPUs.
Storage Utilization	The percentage of provisioned storage capacity currently in use. Represents the total allocated space for all tablespaces.
Sessions	The number of sessions in the database.
Execute Count	The number of user and recursive calls that ran SQL statements during the selected interval.
Running Statements	The number of running SQL statements, aggregated across all consumer groups, during the selected interval.
Queued Statements	The number of queued SQL statements, aggregated across all consumer groups, during the selected interval.
Database Availability	The database is available for connections during the selected time interval (data for this metric lags by 5 minutes). Possible values for this metric: • 1 = Database is Available • 0 = Database is Unavailable You can set an alarm that is triggered if the database is not available (value 0). **Note:** Availability is calculated based on the "Monthly Uptime Percentage" described in the Oracle PaaS and IaaS Public Cloud Services Pillar Document document under **Delivery Policies** (see Autonomous Database Availability Service Level Agreement).

Metric Name	Description
Regional Availability	Shows the average availability percentage of Autonomous Databases across a set of services in each region, ensuring that regions are performing as per Service Level Agreement. Availability data is updated daily for each region.
Failed Connections	Shows the total number of failed connections to the database during the selected interval.

Optionally, to view all the Autonomous Database metrics:

1. In the Metrics area click **View all database metrics**.
2. In the **Metric namespace** dropdown, select **oci_autonomous_database**.
3. Next to **Dimensions**, click **Add**.
4. In the Edit dimensions dialog, for **deploymentType** select **Shared**.

To create an alarm on a metric, click **Options** and select **Create an Alarm on this Query**. See Managing Alarms for information on setting and using alarms.

For more information about metrics see Available Metrics: oci_autonomous_database.

You can also use the Monitoring API to view metrics. See Monitoring API for more information.

- View Logs and Audit Trails
  Shows the steps to view the Autonomous Database logs and audit trails.

View Logs and Audit Trails

Shows the steps to view the Autonomous Database logs and audit trails.

> **Note:**
>
> To view logs and audit trials you must have the required access as specified in an Oracle Cloud Infrastructure policy (whether you're using the Console, the REST API, or another tool). See Getting Started with Policies for information on policies.

To view audit trails and logs for an Autonomous Database instance:

1. On the Details page, under **Resources**, click **Metrics**.
2. In the Metrics area click **View audit and logs**.
3. In the Logging area, click **Logs** to view log information.
4. In the logging area, click **Audit** to view audit information.

See Audit Autonomous Database and Audit Logs for more information.

View Metrics for Autonomous Databases in a Compartment

Shows the steps to view metrics for Autonomous Databases in a compartment.

To view metrics you must have the required access as specified in an Oracle Cloud Infrastructure policy (whether you're using the Console, the REST API, or other tool). See Getting Started with Policies for information on policies.

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the left navigation list click **Observability & Management**. Under **Monitoring**, click **Service Metrics**.

To use the metrics service to view Autonomous Database metrics:

1. On the Service Metrics page, under **Compartment** select your compartment.
2. On the Service Metrics page, under **Metric Namespace** select **oci_autonomous_database**.
3. If there are multiple Autonomous Databases in the compartment you can show metrics aggregated across the Autonomous Databases by selecting **Aggregate Metric Streams**.
4. If you want to limit the metrics you see, next to **Dimensions** click **Add** (click **Edit** if you have already added dimensions).
   a. In the **Dimension Name** field select a dimension.
   b. In the **Dimension Value** field select a value.
   c. Click **Done**.

   In the Edit dimensions dialog click **+Additional Dimension** to add an additional dimension. Click **x** to remove a dimension.

To create an alarm on a specific metric, click **Options** and select **Create an Alarm on this Query**. See Managing Alarms for information on setting and using alarms.

Autonomous Database Metrics and Dimensions

You can limit the instances where you see metrics with dimensions. The available dimensions include: workload type, instance display name, region, and the instance OCID.

Use dimensions by selecting values in the Oracle Cloud Infrastructure Console Service Metrics page or by setting dimension values with the API. See View Metrics for Autonomous Databases in a Compartment to view metrics and to select metric dimensions.

Use Database Management Service to Monitor Databases

You can use Database Management Service to monitor the health of a single Autonomous Database or a fleet of Autonomous Databases.
Database Management Service lets you:

- Monitor the key performance and configuration metrics of a fleet of Autonomous Databases.
- Compare and analyze database metrics over a selected period.
- Group your critical Autonomous Databases, which reside across compartments, into a Database Group, and monitor them.

To enable Database Management Service:

- Perform the prerequisite steps as described in General Prerequisite Tasks.
- Obtain the required permissions as described in Permissions Required to Enable Database Management for Autonomous Databases.
- Obtain additional permissions to use the Database Management **Fleet Summary** and **Managed Database Details** pages and view alarms on the **Fleet Summary** page. These are listed in Additional Permissions Required to Use Database Management.

- Finally, enable Database Management for Autonomous Databases by following the steps outlined in Enable Database Management for Autonomous Databases.

After enabling Database Management Service, you can perform the actions listed below.

Action	Further Reference
Monitor the health of your fleet of Autonomous Databases on the **Fleet Summary** page.	Monitor the Health of Your Oracle Database Fleet Assess the Performance of Your Databases at a Glance
Monitor a single Autonomous Database on the **Managed Database Details** page.	Monitor and Manage an Autonomous Database
Group Autonomous Databases that reside across compartments into a Database Group and monitor them.	Create and Use Database Groups

> **Note:**
>
> Database Management features such as AWR Explorer and Performance Hub on Managed Database Details are *unavailable* for Autonomous Databases. However, Performance Hub for Autonomous Databases continues to be a free feature that you can access from the Autonomous Database Details page without enabling Database Management.
> For further information, see Monitor Autonomous Database with Performance Hub.

Perform SQL Tracing on Autonomous Database

Use SQL tracing to help you identify the source of an excessive database workload, such as a high load SQL statement in your application.

- Configure SQL Tracing on Autonomous Database
  Shows the steps to configure SQL tracing on Autonomous Database.
- Enable SQL Tracing on Autonomous Database
  Shows the steps to enable SQL tracing for the database session.
- Disable SQL Tracing on Autonomous Database
  Shows the steps to disable SQL tracing on Autonomous Database.
- View Trace File Saved to Cloud Object Store on Autonomous Database
  Describes the output file naming for SQL trace files and shows the commands to use `TKPROF` to organize and view trace file data.
- View Trace Data in SESSION_CLOUD_TRACE View on Autonomous Database
  When you enable SQL Tracing, the same trace information that is saved to the trace file on Cloud Object Store is available in the `SESSION_CLOUD_TRACE` view in the session where the tracing was enabled.

Configure SQL Tracing on Autonomous Database

Shows the steps to configure SQL tracing on Autonomous Database.

> **Note:**
>
> If you enable SQL Tracing your application performance for the session may be degraded while the trace collection is enabled. This negative performance impact is expected due to the overhead of collecting and saving trace data.

To configure your database for SQL tracing, do the following:

1. Create a bucket to store trace files in your Cloud Object Storage.

   To save the SQL tracing files, the bucket can be in any Cloud Object Store that Autonomous Database supports.

   For example, to create a bucket in Oracle Cloud Infrastructure Object Storage, do the following

   a. Open the Oracle Cloud Infrastructure Console.

   b. Select **Storage** from the menu.

   c. Under Storage, select **Object Storage and Archive Storage**.

   d. Click **Create Bucket**.

   e. In the Create Bucket page, enter the **Bucket Name** and click **Create**.

   If you are using an Oracle Cloud Infrastructure Object Storage, note that SQL tracing files are only supported with buckets created in the standard storage tier, make sure you pick **Standard** as the storage tier when creating your bucket. For information on the Standard Object Storage Tier, see Overview of Object Storage.

2. Create a credential for your Cloud Object Storage account using `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DEF_CRED_NAME',
       username => 'adb_user@example.com',
       password => 'password'
   );
   END;
   /
   ```

   See CREATE_CREDENTIAL Procedure for details on the arguments for `username` and `password` parameters for different object storage services.

3. Set initialization parameters to specify the Cloud Object Storage URL for a bucket for SQL trace files and to specify the credentials to access the Cloud Object Storage.

   a. Set database property `DEFAULT_LOGGING_BUCKET` to specify the logging bucket on Cloud Object Storage.

For example, if you create the bucket with Oracle Cloud Infrastructure Object Storage:

```
SET DEFINE OFF;
ALTER DATABASE PROPERTY SET
   DEFAULT_LOGGING_BUCKET = 'https://objectstorage.us-
ashburn-1.oraclecloud.com/n/namespace-string/b/bucket_name/o/';
```

Where *namespace-string* is the Oracle Cloud Infrastructure Object Storage namespace and *bucket_name* is the name of the bucket you previously created. See Understanding Object Storage Namespaces for more information.

See Regions and Availability Domains for a list of regions.

The Cloud Object Store you use for SQL Tracing files can be any Cloud Object Store that Autonomous Database supports.

b. Set the database property `DEFAULT_CREDENTIAL` to the credential you created in Step 2.

For example:

```
ALTER DATABASE PROPERTY SET DEFAULT_CREDENTIAL = 'ADMIN.DEF_CRED_NAME';
```

Including the schema name with the credential is required. In this example the schema is "`ADMIN`".

Enable SQL Tracing on Autonomous Database

Shows the steps to enable SQL tracing for the database session.

> **Note:**
>
> If you enable SQL tracing your application performance for the session may be degraded while the trace collection is enabled. This negative performance impact is expected due to the overhead of collecting and saving trace data.

Before you enable SQL tracing you must configure the database to save SQL Trace files. See Configure SQL Tracing on Autonomous Database for more information.

To enable SQL tracing, do the following:

1. (Optional) Set a client identifier for the application. This step is optional but is recommended. SQL tracing uses the client identifier as a component of the trace file name when the trace file is written to Cloud Object Store.

   For example:

   ```
   BEGIN
     DBMS_SESSION.SET_IDENTIFIER('sqlt_test');
   END;
   /
   ```

2. (Optional) Set a module name for the application. This step is optional but is recommended. SQL tracing uses the module name as a component of the trace file name when the trace file is written to Cloud Object Store.

For example:

```
BEGIN
  DBMS_APPLICATION_INFO.SET_MODULE('modname', null);
END;
/
```

3. Enable the SQL Trace facility.

```
ALTER SESSION SET SQL_TRACE = TRUE;
```

4. Run your workload.

   This step involves running the entire application or specific parts of the application. While you run your workload in the database session, SQL tracing data is collected.

5. Disable SQL Tracing.

   When you disable SQL tracing the collected data for the session is written to a table in your session and to a trace file in the bucket you configure when you set up SQL tracing. See Disable SQL Tracing on Autonomous Database for details.

Disable SQL Tracing on Autonomous Database

Shows the steps to disable SQL tracing on Autonomous Database.

To disable SQL tracing, do the following:

1. Disable the SQL Trace facility.

```
ALTER SESSION SET SQL_TRACE = FALSE;
```

2. (Optional) as needed for your environment, you may want to reset the database property `DEFAULT_LOGGING_BUCKET` to clear the value for the logging bucket on Cloud Object Storage.

   For example:

```
ALTER DATABASE PROPERTY SET DEFAULT_LOGGING_BUCKET = '';
```

When you disable SQL tracing, the tracing data collected while the session runs with tracing enabled is copied to a table and sent to a trace file on Cloud Object Store. You have two options to view trace data:

- View and analyze SQL Trace data in the trace file saved to Cloud Object Store. See View Trace File Saved to Cloud Object Store on Autonomous Database for more information.
- View and analyze SQL Trace data saved to the view `SESSION_CLOUD_TRACE`. See View Trace Data in SESSION_CLOUD_TRACE View on Autonomous Database for more information.

View Trace File Saved to Cloud Object Store on Autonomous Database

Describes the output file naming for SQL trace files and shows the commands to use `TKPROF` to organize and view trace file data.

You use SQL trace file data to analyze application performance on Autonomous Database. When you disable SQL trace in your database session, data is written to the Cloud Object Store bucket configured with `DEFAULT_LOGGING_BUCKET`.

The SQL Trace facility writes the trace data collected in the session to Cloud Object Store in the following format:

default_logging_bucket`/sqltrace/`*`clientID`*`/`*`moduleName`*`/sqltrace_`*`numID1_numID2`*`.trc`

The components of the file name are:

- *default_logging_bucket*: is the value of the `DEFAULT_LOGGING_BUCKET` database property. See Configure SQL Tracing on Autonomous Database for more information.
- *`clientID`*: is the client identifier. See Enable SQL Tracing on Autonomous Database for more information.
- *`moduleName`*: is the module name. See Enable SQL Tracing on Autonomous Database for more information.
- *`numID1_numID2`*: are two identifiers that the SQL Trace facility provides. The *`numID1`* and *`numID2`* numeric values uniquely distinguish each trace file name from other sessions using tracing and creating trace files in the same bucket in the Cloud Object Storage.

  When the database service supports parallelism and a session runs a parallel query, the SQL Trace facility can produce multiple trace files with different *`numID1`* and *`numID2`* values.

> **Note:**
>
> When SQL tracing is enabled and disabled multiple times within the same session, each trace iteration generates a separate trace file in Cloud Object Store. To avoid overwriting previous traces that were generated in the session, subsequently generated files follow the same naming convention and add a numeric suffix to the trace file name. This numeric suffix starts with the number 1 and is incremented by 1 for each tracing iteration thereafter.

For example, the following is a sample generated trace file name when you set the client identifier to "`sql_test`" and the module name to "`modname`":

```
sqltrace/sqlt_test/modname/sqltrace_5415_56432.trc
```

You can run `TKPROF` to translate the trace file into a readable output file.

1. Copy the trace file from Object Store to your local system.
2. Navigate to the directory in which the trace file is saved.
3. Run the `TKPROF` utility from the operating system prompt using the following syntax:

```
tkprof filename1 filename2 [waits=yes|no] [sort=option] [print=n]
    [aggregate=yes|no] [insert=filename3] [sys=yes|no] [table=schema.table]
    [explain=user/password] [record=filename4] [width=n]
```

The input and output files are the only required arguments.

4. To view online Help, invoke `TKPROF` without arguments.

See "Tools for End-to-End Application Tracing" in Oracle Database SQL Tuning Guide for information about using the `TKPROF` utility.

View Trace Data in SESSION_CLOUD_TRACE View on Autonomous Database

When you enable SQL Tracing, the same trace information that is saved to the trace file on Cloud Object Store is available in the `SESSION_CLOUD_TRACE` view in the session where the tracing was enabled.

While you are still in the database session you can view SQL tracing data in the `SESSION_CLOUD_TRACE` view. The `SESSION_CLOUD_TRACE` view includes two columns: `ROW_NUMBER` and `TRACE`:

```
DESC SESSION_CLOUD_TRACE

Name       Null? Type
---------- ----- ------------------------------
ROW_NUMBER       NUMBER
TRACE            VARCHAR2(32767)
```

The `ROW_NUMBER` specifies the ordering for trace data found in the `TRACE` column. Each line of trace output written to a trace file becomes a row in the table and is available in the `TRACE` column.

After you disable SQL tracing for the session, you can run queries on the `SESSION_CLOUD_TRACE` view.

For example:

```
SELECT trace FROM SESSION_CLOUD_TRACE ORDER BY row_number;
```

The data in `SESSION_CLOUD_TRACE` persists for the duration of the session. After you log out or close the session, the data is no longer available.

If SQL Trace is enabled and disabled multiple times within the same session, `SESSION_CLOUD_TRACE` shows the trace data for all the iterations cumulatively. Thus, re-enabling tracing in a session after previously disabling tracing does not remove the trace data produced by the earlier iteration.

Service Console Replacement with Database Actions

The Autonomous Database functionality that was available from the Service Console is now available in Database Actions. If you are familiar with the Service Console, the following provides a mapping of equivalent functionality in Database Actions.

Service Console Area	Equivalent in Database Actions	More Information
Overview	Monitoring: Database Monitor → Overview tab	Database Dashboard Overview

Service Console Area	Equivalent in Database Actions	More Information
Activity: Monitor Tab	Monitoring: Database Monitor → Monitor tab	Database Dashboard Activity
Activity: Monitored SQL Tab	Monitoring: Performance Hub	Use Database Actions to Monitor Active Session History Analytics and SQL Statements
Administration: Download Client Credentials (Wallet)	Administration: Download Client Credentials (Wallet)	Download Client Credentials (Wallets)
Administration: Set Resource Management Rules	Administration: Set Resource Management Rules	Manage Concurrency and Priorities on Autonomous Database Manage CPU/IO Shares on Autonomous Database Manage Runaway SQL Statements on Autonomous Database
Administration: Set Administrator Password	Administration: Database Users	Manage the Administrator Account on Autonomous Database
Administration: Manage Oracle ML Users	Development: Oracle Machine Learning	Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database
Development: Download Oracle Instant Client	Downloads: Download Oracle Instant Client	Connect to Autonomous Database Using a Client Application
Development: Download SODA Drivers	Downloads: Download SODA Drivers	Use SODA for REST with Autonomous Database
Development: Oracle APEX	Development: APEX	Access Oracle APEX Administration Services
Development: Database Actions	Database Actions	Connect with Built-In Oracle Database Actions
Development: Oracle Machine Learning User Interface	Development: Oracle Machine Learning	Work with Oracle Machine Learning User Interface for Data Access, Analysis, and Discovery
Development: RESTful Services and SODA	Related Services: RESTful Services and SODA	Access RESTful Services and SODA for REST
Development: Oracle Machine Learning RESTful services	Related Services: Oracle Machine Learning RESTful services	REST API for Oracle Machine Learning Services
Development: Oracle Database API for MongoDB	Related Services: Oracle Database API for MongoDB	Connect MongoDB Applications to Autonomous Database

Available Metrics: oci_autonomous_database

This topic describes the metrics emitted by the Database service in the `oci_autonomous_database` namespace.

Database service metrics for Autonomous Databases include the following **dimensions**:

- **AUTONOMOUSDBTYPE**
  The type of Autonomous Database, Autonomous Data Warehouse (ADW) or Autonomous Transaction Processing (ATP).
- **deploymentType**
  The Exadata infrastructure type, shared or dedicated. When using the Console to view default metric charts for multiple Autonomous Databases, you must specify this dimension.

This topic covers Oracle Autonomous Database Serverless metrics. See Monitor Databases with Autonomous Database Metrics for information on Oracle Autonomous Database on Dedicated Exadata Infrastructure metrics.

- **DISPLAYNAME**
  The friendly name of the Autonomous Database.
- **REGION**
  The **region** in which the Autonomous Database resides.
- **RESOURCEID**
  The **OCID** of the Autonomous Database.
- **RESOURCENAME**
  The name of the Autonomous Database.

The metrics listed in the following table are automatically available for any Autonomous Database that you create. You do not need to enable monitoring on the resource to get these metrics.

> **Note:**
>
> Valid alarm intervals are 5 minutes or greater due to the frequency at which these metrics are emitted. See To create an alarm for details on creating alarms.

In the following table, metrics that are marked with an asterisk (*) can be viewed only on the **Service Metrics** page of the Oracle Cloud Infrastructure console. All metrics can be filtered by the dimensions described in this topic.

Metric	Metric Display Name	Unit	Description	Collection Frequency
`ConnectionLatency`	Connection Latency	Milliseconds	The time taken to connect to an Oracle Autonomous Database Serverless instance in each region from a Compute service virtual machine in the same region. Statistic: Max Interval: 5 minutes	5 minutes
`CpuTime*`	CPU Time	seconds per second	Average rate of accumulation of CPU time by foreground sessions in the database over the time interval. Statistic: Mean Interval: 1 minute	NA

Metric	Metric Display Name	Unit	Description	Collection Frequency
CpuUtilization	CPU Utilization	percent	The CPU usage expressed as a percentage, aggregated across all consumer groups. The utilization percentage is reported with respect to the number of CPUs the database is allowed to use. Statistic: Mean Interval: 1 minute	NA
CurrentLogons*	Current Logons	count	The number of successful logons during the selected interval. Statistic: Count Interval: 1 minute	1 minute
DatabaseAvailability	Database Availability	count	The database is available for connections in the given minute, with possible values: • 1 = DB Available • 0 = DB Unavailable Statistic: Mean Interval: 1 minute	5 minutes

Metric	Metric Display Name	Unit	Description	Collection Frequency
DBBlockChanges	DB Block Changes	count	The number of changes that were part of an update or delete operation that were made to all blocks in the SGA. Such changes generate redo log entries and thus become permanent changes to the database if the transaction is committed. This approximates total database work. This statistic indicates the rate at which buffers are being dirtied, during the selected time interval. Statistic: Sum Interval: 1 minute	1 minute
DBTime*	DB Time	seconds per second	The amount of time database user sessions spend executing database code (CPU Time + WaitTime). DB Time is used to infer database call latency, because DB Time increases in direct proportion to both database call latency (response time) and call volume. It is calculated as the average rate of accumulation of database time by foreground sessions in the database over the time interval. Statistic: Mean Interval: 1 minute	NA

Metric	Metric Display Name	Unit	Description	Collection Frequency
ExecuteCount	Execute Count	count	The number of user and recursive calls that executed SQL statements during the selected interval. Statistic: Sum Interval: 1 minute	NA
FailedConnections*	Failed Connections	count	The number of failed database connections. Statistic: Sum Interval: 1 minute	1 minute
FailedLogons	Failed Logons	count	The number of log ons that failed because of an invalid user name and/or password, during the selected interval. Statistic: Mean Interval: 1 minute	1 minute
HardParseCount	Parse Count (Hard)	count	The number of parse calls (real parses) during the selected time interval. A hard parse is an expensive operation in terms of memory use, because it requires Oracle to allocate a workheap and other memory structures and then build a parse tree. Statistic: Sum Interval: 1 minute	1 minute
LogicalReads	Session Logical Reads	count	The sum of "db block gets" plus "consistent gets", during the selected time interval. This includes logical reads of database blocks from either the buffer cache or process private memory.	1 minute

Metric	Metric Display Name	Unit	Description	Collection Frequency
ParseCount*	Parse Count (Total)	count	The number of hard and soft parses during the selected interval. Statistic: Sum Interval: 1 minute	1 minute
ParseFailureCount	Parse Count (Failures)	count	The number of parse failures during the selected time interval. Statistic: Sum Interval: 1 minute	1 minute
PhysicalReads	Physical Reads	count	The number of data blocks read from disk, during the selected time interval. Statistic: Sum Interval: 1 minute	1 minute
PhysicalReadTotalBytes	Physical Read Total Bytes	count	The size in bytes of disk reads by all database instance activity including application reads, backup and recovery, and other utilities, during the selected time interval. Statistic: Sum Interval: 1 minute	1 minute
PhysicalWrites	Physical Writes	count	The number of data blocks written to disk, during the selected time interval. Statistic: Sum Interval: 1 minute	1 minute
PhysicalWriteTotalBytes	Physical Write Total Bytes	count	The size in bytes of all disk writes for the database instance including application activity, backup and recovery, and other utilities, during the selected time interval. Statistic: Sum Interval: 1 minute	1 minute

Metric	Metric Display Name	Unit	Description	Collection Frequency
QueryLatency	Query Latency	Milliseconds	The time taken to display the results of a simple query on the user's screen. Statistic: Max Interval: 5 minutes	5 minutes
QueuedStatements	Queued Statements	count	The number of queued SQL statements, aggregated across all consumer groups, during the selected interval. Statistic: Sum Interval: 1 minute	NA
RedoGenerated	Redo Generated	count	Amount of redo generated in bytes, during the selected time interval. Statistic: Sum Interval: 1 minute	1 minute
RunningStatements	Running Statements	count	The number of running SQL statements, aggregated across all consumer groups, during the selected interval. Statistic: Mean Interval: 1 minute	1 minute
Sessions	Sessions	count	The number of sessions in the database. Statistic: Sum Interval: 1 minute	1 minute
SQLNetBytesFromClient	Bytes Received via SQL*Net from Client	count	The number of bytes received from the client over Oracle Net Services, during the selected time interval. Statistic: Sum Interval: 1 minute	1 minute

Metric	Metric Display Name	Unit	Description	Collection Frequency
SQLNetBytesFromDBLink	Bytes Received via SQL*Net from DBLink	count	The number of bytes received from a database link over Oracle Net Services, during the selected time interval. Statistic: Sum Interval: 1 minute	1 minute
SQLNetBytesToClient	Bytes Sent via SQL*Net to Client	count	The number of bytes sent to the client from the foreground processes, during the selected time interval. Statistic: Sum Interval: 1 minute	1 minute
SQLNetBytesToDBLink	Bytes Sent via SQL*Net to DBLink	count	The number of bytes sent over a database link, during the selected time interval. Statistic: Sum Interval: 1 minute	1 minute
StorageAllocated*	Storage Space Allocated	GB	Amount of space allocated to the database for all tablespaces, during the interval. Statistic: Max Interval: 1 hour	1 hour
StorageMax	Maximum Storage Space	GB	Maximum amount of storage reserved for the database during the interval. Statistic: Max Interval: 1 hour	1 hour
StorageUsed*	Storage Space Used	GB	Maximum amount of space used during the interval. Statistic: Max Interval: 1 hour	1 hour

Metric	Metric Display Name	Unit	Description	Collection Frequency
StorageUtilization	Storage Utilization	percent	The percentage of the reserved maximum storage currently allocated for all database tablespaces. Represents the total reserved space for all tablespaces. Statistic: Mean Interval: 1 hour	1 hour
TransactionCount*	Transaction Count	count	The combined number of user commits and user rollbacks during the selected interval. Statistic: Sum Interval: 1 minute	1 minute
UserCalls*	User Calls	count	The combined number of logons, parses, and execute calls during the selected interval. Statistic: Sum Interval: 1 minute	1 minute
UserCommits	User Commits	count	The number of user commits during the selected time interval. When a user commits a transaction, the generated redo that reflects the changes made to database blocks must be written to disk. Commits often represent the closest thing to a user transaction rate. Statistic: Sum Interval: 1 minute	1 minute

Metric	Metric Display Name	Unit	Description	Collection Frequency
UserRollbacks	User Rollbacks	count	Number of times users manually issue the ROLLBACK statement or an error occurs during a user's transactions, during the selected time interval. Statistic: Sum Interval: 1 minute	1 minute
WaitTime*	Wait Time	seconds per second	Average rate of accumulation of non-idle wait time by foreground sessions in the database over the time interval. Statistic: Mean Interval: 1 minute	1 minute

Use Sample Data Sets in Autonomous Database

For users who want to start using the service without creating their own tables, Autonomous Database provides the read-only Sales History and Star Schema Benchmark data sets.

These data sets are provided as Oracle Database schemas SH and SSB respectively. Any user can query these data sets without any manual configuration.

Notes for the Sample Data sets:

- Both the SH and SSB are provided as schema-only users, so you cannot unlock or drop those users or set a password.
- The storage size of the sample data sets is not counted towards or billed for in your database's storage.

Sales History (SH) Schema

The SH schema provides a small data set you can use to run the sample queries in the *Oracle Database Data Warehousing Guide*. Note that you need to prefix the table names with the schema name SH in your queries. For example, the following query shows you how the SQL function RANK() works:

```
SELECT channel_desc, TO_CHAR(SUM(amount_sold), '9,999,999,999') SALES$,
   RANK() OVER (ORDER BY SUM(amount_sold)) AS default_rank,
   RANK() OVER (ORDER BY SUM(amount_sold) DESC NULLS LAST) AS custom_rank
FROM sh.sales, sh.products, sh.customers, sh.times, sh.channels, sh.countries
WHERE sales.prod_id=products.prod_id AND sales.cust_id=customers.cust_id
  AND customers.country_id = countries.country_id AND
sales.time_id=times.time_id
  AND sales.channel_id=channels.channel_id
```

```
  AND times.calendar_month_desc IN ('2000-09', '2000-10')
  AND country_iso_code='US'
GROUP BY channel_desc;
```

For more information on the SH schema see *Sample Schemas* and *Schema Diagrams*.

Star Schema Benchmark (SSB) Schema

The SSB schema provides a well-known large sample data set. The SSB schema contains 1 TB of data (storage of the sample data sets does not count towards your database storage). You can use this schema to test the performance of your service. You can run the sample queries on this schema with different database services, HIGH, MEDIUM, LOW and with different number of ECPUs (OCPUs if your database uses OCPUs) to test the performance of Autonomous Database.

The SSB schema contains the tables: `lineorder`, `customer`, `supplier`, `part`, and `dwdate`. See Sample Star Schema Benchmark (SSB) Queries and Analytic Views for a list of sample queries you can use against the SSB schema. Note that you need to prefix the table names with the schema name `SSB` in your queries.

For more information on database services, see Predefined Database Service Names for Autonomous Database.

Use the Oracle Autonomous Database Free Container Image

Use the Oracle Autonomous Database Free Container Image to run Autonomous Database in a container in your own environment, without requiring access to the Oracle Cloud Infrastructure Console or to the internet.

- About the Autonomous Database Free Container Image
  You can access the Oracle Autonomous Database Free Container Image from a repository and run it in your local environment.
- Container Registry Locations for Oracle Autonomous Database Free Container Image
  There are multiple locations where you can obtain Oracle Autonomous Database Free Container Image, including: Oracle Cloud Infrastructure Registry (Container Registry) and GitHub.
- Start Autonomous Database Free Container Image
  After you download Oracle Autonomous Database Free Container Image, you can start the image in a container.
- Perform Database Operations using adb-cli
  The `adb-cli` command-line utility can be used to perform database operations after the container is up and running.
- Connect to an Autonomous Database Free Container Image
  When the Autonomous Database Free Container Image is running in a container, you can connect to an Autonomous Database instance.
- Migrate Data Between Autonomous Database Free Containers

About the Autonomous Database Free Container Image

You can access the Oracle Autonomous Database Free Container Image from a repository and run it in your local environment.

Autonomous Database provides a fully managed Oracle Database that is available on Oracle Cloud Infrastructure. On Oracle Cloud Infrastructure, you perform lifecycle management operations and run Autonomous Database using the Oracle Cloud Infrastructure Console and you connect to your database through the public internet or through a private network that you set up (depending on your network configuration and security needs).

> **Note:**
>
> Autonomous Database supports both the Oracle Autonomous Database Free Container Image 19c and Oracle Autonomous Database Free Container Image 23ai versions.

The Oracle Autonomous Database Free Container Image provides an alternative to run Autonomous Database in a container in your own environment, without requiring access to Oracle Cloud Infrastructure Console or to the internet. When you run Autonomous Database in a container, the container provides a local, isolated environment with additional options for development, testing, and exploration of Oracle Autonomous Database features.

- Oracle Autonomous Database Free Container Image License
  Describes licensing for Oracle Autonomous Database Free Container Image.
- Oracle Autonomous Database Free Container Image Features
  The Oracle Autonomous Database Free Container Image provides many of the features available with Autonomous Database Serverless.
- Oracle Autonomous Database Free Container Image Recommendations and Restrictions
  Describes requirements and restrictions for Autonomous Database Free Container Image.

Oracle Autonomous Database Free Container Image License

Describes licensing for Oracle Autonomous Database Free Container Image.

Oracle Autonomous Database Free Container Image is subject to licensing.

This Licensing Information document is specifically for the Oracle Database in Oracle Autonomous Database Free Container Image. This document is a part of the product or program documentation under the terms of your Oracle license agreement and is intended to help you understand the program editions, entitlements, restrictions, prerequisites, special license rights, and/or separately licensed third party technology terms associated with the Oracle software program(s) covered by this document (the "Program(s)"). Information for other Oracle Database editions (such as Enterprise Edition) will be added to this document once they are released. Entitled or restricted use products or components identified in this document that are not provided with the particular Program may be obtained from the Oracle Software Delivery Cloud website (https://edelivery.oracle.com) or from media Oracle may provide. If you have a question about your license rights and obligations, please contact your Oracle sales representative, review the information provided in Oracle's Software Investment Guide (http://www.oracle.com/us/corporate/pricing/software-investment-guide/index.html), and/or contact the applicable Oracle License Management Services representative listed on http://www.oracle.com/us/corporate/license-management-services/index.html.

See Database Licensing Information User Manual for more information.

Oracle Autonomous Database Free Container Image Features

The Oracle Autonomous Database Free Container Image provides many of the features available with Autonomous Database Serverless.

- Each Autonomous Database Free Container Image provides two Autonomous Database instances, one instance with Data Warehouse workload type and one instance with Transaction Processing workload type.
- The database will be started with either the Transaction Processing workload type or the Data Warehouse workload type based on the workload type you specify during startup.
- You can perform database operations using the `adb-cli` command-line utility.
- The Autonomous Database Free Container Image resource allocation is 4 ECPUs and 20 GB of storage, and allows a maximum of 30 simultaneous database sessions.
- Each Autonomous Database Free Container Image supports the Autonomous Database consumer groups:
  - Data Warehouse workload: You connect through HIGH, MEDIUM, or LOW services
  - Transaction Processing workload: You connect through HIGH, MEDIUM, LOW, TP, or TPURGENT services

  See About Autonomous Database Workload Types for more information.
- The Autonomous Database Free Container Image provides support for the following built-in database tools:
  - Database Actions
  - Oracle REST Data Services (ORDS)
  - Oracle APEX
  - Oracle Database API for MongoDB

Oracle Autonomous Database Free Container Image Recommendations and Restrictions

Describes requirements and restrictions for Autonomous Database Free Container Image.

Recommendations for Resource Allocation for the Autonomous Database Free Container Image

The following is the recommended resource allocation for the Autonomous Database Free Container Image:

- 4 CPUs
- 8 GB memory

Restrictions for Autonomous Database Free Container Image

- There is no automatic patching or maintenance windows for the Autonomous Database Free Container Image. The repository provides the latest version of the Autonomous Database Free Container Image. Check the repository to find newer versions of Autonomous Database Free Container Image.
- The following Autonomous Database built-in tools are not supported:
  - Graph

  - Oracle Machine Learning
  - Data Transforms

- When Autonomous Database runs in a container, the container provides a local Autonomous Database instance. The container image does not include the features that are only available through the Oracle Cloud Infrastructure Console or the APIs. Some features that are available in-database and also available through the Oracle Cloud Infrastructure Console can still be used through in-database commands, such as resetting the `ADMIN` password. The following lists some of the features that are not available:

Feature	Available or Not Available
Backup an Instance	Not available
Choose a Character Set	Not available
Clone an Instance	Not available
Create an Elastic Pool	Not available
Customer-managed keys	Not available
Database rename	Not available
Data Safe	Not available
Disable compute auto scaling	Not available
Disable Built-in Database Tools	Not available
Disable storage auto scaling	Not available
Disaster recovery options, including Autonomous Data Guard and Backup-Based Disaster Recovery.	Not available
Download wallet	Not available
Enable Built-in Database Tools	Not available
Enable compute auto scaling	Not available
Enable storage auto scaling	Not available
Join an Elastic Pool	Not available
Network ACLs	Not available
Oracle Cloud Infrastructure events	Not available
Performance hub	Not available
Private endpoints	Not available
Real Application Testing	Not available
Resource Principal based authentication	Not available
Restart Instance	Not available
Restore an Instance	Not available
Rotate wallet	Not available
Sample Schema	Not available
Scale down CPU and storage	Not available
Scale up CPU and storage	Not available
Selecting the Instance Patch Level	Not available
Start Instance	Not available
Stop Instance	Not available

> **Note:**
>
> When you run Autonomous Database Free Container Image in a container, you can start an instance, stop an instance, or restart an instance by starting, stopping or restarting the container.

Container Registry Locations for Oracle Autonomous Database Free Container Image

There are multiple locations where you can obtain Oracle Autonomous Database Free Container Image, including: Oracle Cloud Infrastructure Registry (Container Registry) and GitHub.

You can obtain the Oracle Autonomous Database Free Container Image in multiple locations. The examples shown use `podman` commands (see Podman for more information).

1. Start a podman virtual machine.

   For example:

   ```
   podman machine init
   podman machine set --cpus 4 --memory 8192
   podman machine start
   ```

2. Obtain the Autonomous Database Free Container Image.

   Oracle Cloud Infrastructure Registry:

   For example, use `podman` command to pull the latest Autonomous Database Free Container Image image:

   ```
   podman pull container-registry.oracle.com/database/adb-free:latest-23ai
   ```

   > **Note:**
   >
   > Autonomous Database Serverless (ADB- S) also supports 19c container images.
   >
   > - For 19c container image names, specify `latest` tag as your database version.
   > - In the above example, to pull the Autonomous Database Free Container Image 19c , use the following command in the image name tag:
   >
   >   ```
   >   podman pull container-registry.oracle.com/database/adb-free:latest
   >   ```
   >
   > Throughout the document, the image name tag refers to the `latest-23ai` version.

   For more details and additional information, search for "Oracle Autonomous Database Free" on Oracle Cloud Infrastructure Registry.

   GitHub Packages:

For example, use `podman` command to pull the Autonomous Database Free Container Image from: GitHub Packages:

```
podman pull ghcr.io/oracle/adb-free:latest-23ai
```

3. Verify the image.

   For example:

```
podman images container-registry.oracle.com/database/adb-free:latest-23ai
```

Start Autonomous Database Free Container Image

After you download Oracle Autonomous Database Free Container Image, you can start the image in a container.

The database will be started with either the Transaction Processing workload type or the Data Warehouse workload type based on the workload type you specify.

- Start a container to run Autonomous Database Free Container Image.

  For example, with podman:

```
podman run -d \
-p 1521:1522 \
-p 1522:1522 \
-p 8443:8443 \
-p 27017:27017 \
-e WORKLOAD_TYPE='ATP' \
-e WALLET_PASSWORD=*** \
-e ADMIN_PASSWORD=*** \
--cap-add SYS_ADMIN \
--device /dev/fuse \
--name adb-free \
container-registry.oracle.com/database/adb-free:latest-23ai
```

  Command notes:

  - Autonomous Database Serverless (ADB- S) also supports 19c container images.
    - For 19c container image names, specify `latest` tag as your database version.
    - In the above example, to start a container to run the Autonomous Database Free Container Image 19c, use the following command in the image name tag:

```
container-registry.oracle.com/database/adb-free:latest
```

  - Throughout the document, the image name tag refers to the `latest-23ai` version.
  - The `WORKLOAD_TYPE` can be either `ATP` or `ADW`. The default value is `ATP`.
  - By default, the database is named either `MYATP` or `MYADW`, depending on the passed `WORKLOAD_TYPE` value. Optionally, if you want a different database name than the default, you can set the `DATABASE_NAME` parameter. The database name can contain only alphanumeric characters.
  - On container startup, the corresponding `MY<WORKLOAD_TYPE>.pdb` is downloaded and plugged in from a public Object Storage bucket.

- The wallet is generated using the provided `WALLET_PASSWORD`.
- It is necessary to change the `ADMIN_PASSWORD` upon initial login.
- Ensure you check the following requirements when you create or modify the `ADMIN_PASSWORD`:
  - The password must be between 12 and 30 characters long and must include at least one uppercase letter, one lowercase letter, and one numeric character.
  - The password cannot contain the username.
- Ensure you check the following requirements when you create or modify the `WALLET_PASSWORD`:
  - The password must be between 8 and 30 characters long and include alphabetic characters combined with numeric characters or special characters.
- For an OFS mount, the container starts with `SYS_ADMIN` capability. Also, virtual device `/dev/fuse` must be accessible.
- This `-p` options specify that the following ports are forwarded to the container process:

Port	Description
1521	TLS
1522	mTLS
8443	HTTPS port for ORDS / APEX and Database Actions
27017	Mongo API

If you are behind a corporate proxy, include `-e` options to specify environment variables for proxies. For example, with podman:

```
podman run -d \
-p 1521:1522 \
-p 1522:1522 \
-p 8443:8443 \
-p 27017:27017 \
-e WORKLOAD_TYPE='ATP' \
-e WALLET_PASSWORD=*** \
-e ADMIN_PASSWORD=*** \
-e http_proxy=http://example-corp-proxy.com:80/ \
-e https_proxy=http://example-corp-proxy.com:80/ \
-e no_proxy=localhost,127.0.0.1 \
-e HTTP_PROXY=http://example-corp-proxy.com:80/  \
-e HTTPS_PROXY=http://example-corp-proxy.com:80/  \
-e NO_PROXY=localhost,127.0.0.1 \
--cap-add SYS_ADMIN \
--device /dev/fuse \
--name adb-free \
container-registry.oracle.com/database/adb-free:latest-23ai
```

Perform Database Operations using `adb-cli`

The `adb-cli` command-line utility can be used to perform database operations after the container is up and running.

To use `adb-cli`, you can define the following alias for convenience:

```
alias adb-cli="podman exec <container_name> adb-cli"
```

Available Commands

You can view the list of available commands using the following command:

```
adb-cli --help
Usage: adb-cli [OPTIONS] COMMAND [ARGS]...
    ADB-S Command Line Interface (CLI) to perform container-runtime database
operations
Options:
    -v, --version Show the version and exit.
    --help Show this message and exit.
Commands:
    add-database
    change-password
```

Add Database

You can add a database using the following command:

```
adb-cli add-database --workload-type "ATP" --admin-password
"Welcome_MY_ATP_1234"
```

Change Password

You can change the admin password using the following command:

```
adb-cli change-password --database-name "MYATP"
    --old-password "Welcome_MY_ATP_1234" --new-password "Welcome_12345"
```

Connect to an Autonomous Database Free Container Image

When the Autonomous Database Free Container Image is running in a container, you can connect to an Autonomous Database instance.

To connect to an Autonomous Database instance running in a container, you can either setup the wallet and connect or use a TLS walletless connection.

- ORDS/APEX/Database Actions
  The container `hostname` is used to generate self-signed SSL certificates to serve HTTPS traffic on port 8443. Oracle APEX and Database Actions can be accessed using the container host (or simply `localhost`).
- Available TNS Aliases
  You can use any one of the following aliases to connect to Autonomous Database free container.

- Setup a Wallet and Connect
- Setup TLS Walletless Connection and Connect

ORDS/APEX/Database Actions

The container `hostname` is used to generate self-signed SSL certificates to serve HTTPS traffic on port 8443. Oracle APEX and Database Actions can be accessed using the container host (or simply `localhost`).

Application	URL
Oracle APEX	https://localhost:8443/ords/apex
Database Actions	https://localhost:8443/ords/sql-developer

> **Note:**
>
> For additional databases plugged in using `adb-cli add-database` command, use the URL formats `https://localhost:8443/ords/{database_name}/apex` and `https://localhost:8443/ords/{database_name}/sql-developer` to access APEX and Database Actions respectively.

Available TNS Aliases

You can use any one of the following aliases to connect to Autonomous Database free container.

Table 3-5 Available TNS Aliases

Protocol	Transaction Processing Workload	Data Warehouse Workload
mTLS	• myatp_medium • myatp_high • myatp_low • myatp_tp • myatp_tpurgent	• myadw_medium • myadw_high • myadw_low
TLS	• myatp_medium_tls • myatp_high_tls • myatp_low_tls • myatp_tp_tls • myatp_tpurgent_tls	• myadw_medium_tls • myadw_high_tls • myadw_low_tls

The TNS alias mappings for these connect strings are in `$TNS_ADMIN/tnsnames.ora`. See Manage Concurrency and Priorities on Autonomous Database for information on the service names in `tnsnames.ora`.

Setup a Wallet and Connect

Perform the following steps to setup a wallet and connect:

1. Copy the generated wallet to your host.

When you start the container, Autonomous Database generates a wallet in `/u01/app/oracle/wallets/tls_wallet`.

For example, copy the generated wallet to the local folder `/scratch/tls_wallet`:

```
podman cp adb-free:/u01/app/oracle/wallets/tls_wallet /scratch/tls_wallet
```

This copies the wallet to the folder: `/scratch/tls_wallet`.

2. Set the value of the `TNS_ADMIN` environment variable to the wallet directory.

   For example:

   ```
   export TNS_ADMIN=/scratch/tls_wallet
   ```

3. If you want to connect to a remote host where the Autonomous Database Free Container Image is running, replace `localhost` in `$TNS_ADMIN/tnsnames.ora` with the remote host FQDN.

   For example:

   ```
   sed -i 's/localhost/example.com/g' $TNS_ADMIN/tnsnames.ora
   ```

4. Connect to the Autonomous Database instance.

   For example use `sqlplus` to connect to the **Transaction Processing** workload Autonomous Database instance:

   ```
   sqlplus admin/password@myatp_low_tls
   ```

   For example, use `sqlplus` to connect to the **Data Warehouse** workload Autonomous Database instance:

   ```
   sqlplus admin/password@myadw_low_tls
   ```

Setup TLS Walletless Connection and Connect

To connect without a wallet, you need to update your client's truststore with the self-signed certificate generated at container start.

Perform the following steps to setup a TLS walletless connection and connect.

1. Copy `/u01/app/oracle/wallets/tls_wallet/adb_container.cert` from the container and update your system truststore.

   For example:

   ```
   podman cp adb-free:/u01/app/oracle/wallets/tls_wallet/adb_container.cert
   adb_container.cert
   sudo cp adb_container.cert /etc/pki/ca-trust/source/anchors
   sudo update-ca-trust
   ```

2. Connect to the Autonomous Database instance.

For example, use `sqlplus` to connect to the Transaction Processing workload Autonomous Database instance:

```
sqlplus admin/password@myatp_low
```

For example, use `sqlplus` to connect to the Data Warehouse workload Autonomous Database instance:

```
sqlplus admin/password@myadw_low
```

Migrate Data Between Autonomous Database Free Containers

When a new version of the Autonomous Database Free Container Image is available, you can migrate data from a container to another container.

For example, use existing data that you created in a container by migrating that data to the latest version of Autonomous Database Free Container Image when a new update is available.

1. Create a podman volume.

   For example:

   ```
   podman volume create adb_container_volume
   ```

2. Verify the volume mountpoint.

   The mountpoint is a podman managed directory location.

   ```
   podman inspect adb_container_volume

   [
        {
             "Name": "adb_container_volume",
             "Driver": "local",
             "Mountpoint": "/share/containers/storage/volumes/
   adb_container_volume/_data",
             "CreatedAt": "2023-09-11T21:23:34.305877073Z",
             "Labels": {},
             "Scope": "local",
             "Options": {},
             "MountCount": 0,
             "NeedsCopyUp": true,
             "NeedsChown": true
        }
   ]
   ```

3. Start the source container, mounting the volume to `/u01/data` in the container.

   For example:

   ```
   podman run -d \
   -p 1521:1522 \
   -p 1522:1522 \
   -p 8443:8443 \
   -p 27017:27017 \
   -e WORKLOAD_TYPE='ATP' \
   ```

```
-e WALLET_PASSWORD=*** \
-e ADMIN_PASSWORD=*** \
--cap-add SYS_ADMIN \
--device /dev/fuse \
--name source_adb_container \
--volume adb_container_volume:/u01/data \
container-registry.oracle.com/database/adb-free:latest-23ai
```

> **Note:**
>
> Autonomous Database Serverless (ADB- S) also supports 19c container images.
>
> - For 19c container image names, specify `latest` tag as your database version.
> - In the above example, to start the Autonomous Database Free Container Image 19c , use the following command in the image name tag:
>
>   `container-registry.oracle.com/database/adb-free:`**`latest`**
>
> Throughout the document, the image name tag refers to the `latest-23ai` version.

4. This example assumes you have previously created your data in schema, `app_user`.
5. Export the data from `app_user`'s schema to the container volume.
   a. Connect as `ADMIN` and create `ORA_EXP_DIR` directory pointing to `/u01/data`.

```
sqlplus admin/**************@myadw_high

SQL> exec DBMS_CLOUD_CONTAINER_ADMIN.create_export_directory('/u01/
data');


SQL> select directory_path from dba_directories where
directory_name='ORA_EXP_DIR';

DIRECTORY_PATH
--------------------------------------------------------------------------------
--------
/u01/data
```

   b. Run the export job in schema mode and for both `ADMIN` and `APP_USER` schemas.

```
SET scan off
SET serveroutput ON
SET escape off

DECLARE
    h1 NUMBER;
    s VARCHAR2(1000):=NULL;
    errorvarchar VARCHAR2(100):= 'ERROR';
    tryGetStatus NUMBER := 0;
    success_with_info EXCEPTION;
```

```
    PRAGMA EXCEPTION_INIT(success_with_info, -31627);
BEGIN
    h1 := dbms_datapump.OPEN(operation => 'EXPORT', job_mode =>
'SCHEMA', job_name => 'EXPORT_MY_ADW_4', version => 'COMPATIBLE');
    tryGetStatus := 1;
    dbms_datapump.add_file(handle => h1, filename =>
'EXPORT_MY_ADW.LOG', directory => 'ORA_EXP_DIR', filetype =>
DBMS_DATAPUMP.KU$_FILE_TYPE_LOG_FILE);
    dbms_datapump.metadata_filter(handle => h1, name => 'SCHEMA_EXPR',
VALUE => 'IN(''ADMIN'', ''APP_USER'')');
    dbms_datapump.add_file(handle => h1, filename => 'MY_ADW_%U.DMP',
directory => 'ORA_EXP_DIR', filesize => '500M',  filetype =>
DBMS_DATAPUMP.KU$_FILE_TYPE_DUMP_FILE);
    dbms_datapump.start_job(handle => h1, skip_current => 0, abort_step
=> 0);
    dbms_datapump.detach(handle => h1);
    errorvarchar := 'NO_ERROR';
EXCEPTION
    WHEN OTHERS THEN
    BEGIN
        IF ((errorvarchar = 'ERROR')AND(tryGetStatus=1)) THEN
            DBMS_DATAPUMP.DETACH(h1);
        END IF;
    EXCEPTION
    WHEN OTHERS THEN
        NULL;
    END;
    RAISE;
END;
/
```

6. Verify the export.

   List the files in the container `/u01/data`.

   ```
   podman exec -it source_adb_container bash
   cd /u01/data
   ```

   Verify the export log (`export log`), check for any errors and successful completion.

7. Stop and remove the source container.

   ```
   podman stop source_adb_container
   podman rm source_adb_container
   ```

   > **Note:**
   >
   > The `adb_container_volume` will live across container restarts and removals

8. Start a destination container mounting the same volume to `/u01/data` in the container.

   ```
   podman run -d \
   -p 1521:1522 \
   -p 1522:1522 \
   ```

```
-p 8443:8443 \
-p 27017:27017 \
-e WORKLOAD_TYPE='ATP' \
-e WALLET_PASSWORD=*** \
-e ADMIN_PASSWORD=*** \
--cap-add SYS_ADMIN \
--device /dev/fuse \
--name dest_adb_container \
--volume adb_container_volume:/u01/data \
container-registry.oracle.com/database/adb-free:latest-23ai
```

9. Import data in destination container.

   Connect as `ADMIN` and create `ORA_EXP_DIR` directory pointing to `/u01/data`.

```
SQL> exec DBMS_CLOUD_CONTAINER_ADMIN.create_export_directory('/u01/data');

PL/SQL procedure successfully completed.

SQL> select directory_path from dba_directories where
directory_name='ORA_EXP_DIR';

DIRECTORY_PATH
--------------------------------------------------------------------------------
-----
/u01/data
```

10. Run the import PL/SQL commands.

```
SET scan off
SET serveroutput ON
SET escape off

DECLARE
    h1 NUMBER;
    s VARCHAR2(1000):=NULL;
    errorvarchar VARCHAR2(100):= 'ERROR';
    tryGetStatus NUMBER := 0;
    success_with_info EXCEPTION;
    PRAGMA EXCEPTION_INIT(success_with_info, -31627);
BEGIN
    h1 := dbms_datapump.OPEN(operation => 'IMPORT', job_mode => 'SCHEMA',
job_name => 'IMPORT_MY_ADW_4', version => 'COMPATIBLE');
    tryGetStatus := 1;
    dbms_datapump.add_file(handle => h1, filename => 'IMPORT_MY_ADW.LOG',
directory => 'ORA_EXP_DIR', filetype =>
DBMS_DATAPUMP.KU$_FILE_TYPE_LOG_FILE);
    dbms_datapump.metadata_filter(handle => h1, name => 'SCHEMA_EXPR',
VALUE => 'IN(''ADMIN'', ''APP_USER'')');
    dbms_datapump.add_file(handle => h1, filename => 'MY_ADW_%U.DMP',
directory => 'ORA_EXP_DIR', filesize => '500M',  filetype =>
DBMS_DATAPUMP.KU$_FILE_TYPE_DUMP_FILE);
    dbms_datapump.start_job(handle => h1, skip_current => 0, abort_step =>
0);
    dbms_datapump.detach(handle => h1);
    errorvarchar := 'NO_ERROR';
```

```
EXCEPTION
    WHEN OTHERS THEN
    BEGIN
        IF ((errorvarchar = 'ERROR')AND(tryGetStatus=1)) THEN
            DBMS_DATAPUMP.DETACH(h1);
        END IF;
    EXCEPTION
    WHEN OTHERS THEN
        NULL;
    END;
    RAISE;
END;
/
```

11. Verify the import log.
12. Verify the import data.

4
Tutorials

- Load and Analyze Your Data with Autonomous Database
  Learn about Autonomous Database Serverless and learn how to create an Autonomous Database in just a few clicks. Then load data into your database, query it, and visualize it.
- Self-service Tools for Everyone Using Oracle Autonomous Database
  Introduces the suite of data tools built into Oracle Autonomous Database.
- Manage and Monitor Autonomous Database
  Connect using secure wallets and monitor your Autonomous Database instances.
- Integrate, Analyze and Act on All Data using Autonomous Database
- Load the Autonomous Database with Data Studio
- QuickStart Java applications with Autonomous Database
  Follow these easy steps to connect your Java applications to Autonomous Database using the Oracle JDBC driver and Universal Connection Pool.

Load and Analyze Your Data with Autonomous Database

Learn about Autonomous Database Serverless and learn how to create an Autonomous Database in just a few clicks. Then load data into your database, query it, and visualize it.

 Load and analyze data in Autonomous Database Workshop

- Provision a new Autonomous Database instance.
- Run Queries on the sample data sets
- Upload files to the Oracle Cloud Infrastructure Object Storage, create sample tables, load data into them from files on the OCI Object Storage
- Query files on the Oracle Cloud Infrastructure Object Storage directly without loading them to your database
- Visualize your data using Oracle Analytics Desktop

Self-service Tools for Everyone Using Oracle Autonomous Database

Introduces the suite of data tools built into Oracle Autonomous Database.

 Data Studio - Self-service tools for everyone using Oracle Autonomous Database

This is an overview Data Studio workshop to introduce you to different tools using a sample data analysis task. After this workshop you will have a working knowledge on how to use these tools.

The tools covered are:

- Catalog

- Data Load
- Data Transforms
- Data Analysis
- Data Insight

After completing this workshop, you can look at the individual in-depth workshops for each tool.

Manage and Monitor Autonomous Database

Connect using secure wallets and monitor your Autonomous Database instances.

Manage and Monitor Autonomous Database Workshop

- Get comfortable with Oracle's public cloud services
- Provision a new Autonomous Database instance Serverless
- Run sample queries against the sample data sets
- Load data from the object store
- Query external data from the object store
- Scale an Autonomous Database instance

Integrate, Analyze and Act on All Data using Autonomous Database

This workshop uses Oracle Cloud's modern data platform for analytics to enable MovieStream to better understand their customers, predict churn and provide targeted recommendations and promotions.

- MovieStream stores its data across Oracle Object Storage and Autonomous Database.
- Data is captured from various sources into a landing zone in object storage. This data is then processed, cleansed, transformed, and optimized, and stored in a gold zone on object storage.
- After the data is curated, it is loaded into an Autonomous Database where users can take advantage of a variety of Autonomous Database capabilities: advanced Oracle SQL to glean customer insights, Oracle Machine Learning to predict churn, and Oracle Spatial and Graph to provide targeted promotions.

Integrate, Analyze and Act on All Data using Autonomous Database

Load the Autonomous Database with Data Studio

Data Studio is built into every Autonomous Database, and provides tools to load, transform and prepare data from a wide range of source systems.

This workshop provides practical examples of how to load data from several different systems, as well as how to link to, and feed data in from, data in cloud storage systems.

Load the Autonomous Database with Data Studio

QuickStart Java applications with Autonomous Database

Follow these easy steps to connect your Java applications to Autonomous Database using the Oracle JDBC driver and Universal Connection Pool.

QuickStart Java applications with Oracle Autonomous Database

5

Features

- Using Data Lake Capabilities with Autonomous Database
  Provides information on using Autonomous Database as a data lakehouse.
- The Data Studio Overview Page
  The Data Studio comprises of the Data Load, the Data Analysis, the Data Insights, Catalog and the Data Share tool.
- High Availability
  You can configure and manage Autonomous Data Guard and maintain continuous availability using disaster recovery.
- Security
  Autonomous Database stores all data with inbuilt security. Only authenticated users and applications can access the data when they connect to the database.
- Performance Monitor and Management
  Oracle Autonomous Database Serverless includes several features that automatically monitor, analyze and optimize the performance of your Autonomous Database.
- Using Oracle Graph with Autonomous Database
  Oracle Graph with Autonomous Database enables you to create graphs from data in your Autonomous Database. With graphs you can analyze your data based on connections and relationships between data entities.
- Use Oracle Machine Learning with Autonomous Database
- Creating Analytic Views
  You can create Analytic Views and view information about them. You can also edit and perform other actions on them.
- Use Select AI to Generate SQL from Natural Language Prompts
- JSON Document Stores
  Autonomous Database has full support for data represented as JSON documents. In Autonomous Databases, JSON documents can coexist with relational data.
- Cache Messages with Singleton Pipes and Using Pipes for Persistent Messaging
- Query Text in Object Storage
  The PL/SQL package `DBMS_CLOUD` enables you to build a text index on the object store files, which allows you to search the text and use wildcards with your search.
- Use Oracle Workspace Manager on Autonomous Database
  Workspace Manager provides an infrastructure that enables applications to create workspaces and group different versions of table row values in different workspaces.
- Use and Manage Elastic Pools on Autonomous Database
- Migrate Oracle Databases to Autonomous Database

Using Data Lake Capabilities with Autonomous Database

Provides information on using Autonomous Database as a data lakehouse.

About SQL Analytics on Data Lakes with Autonomous Database

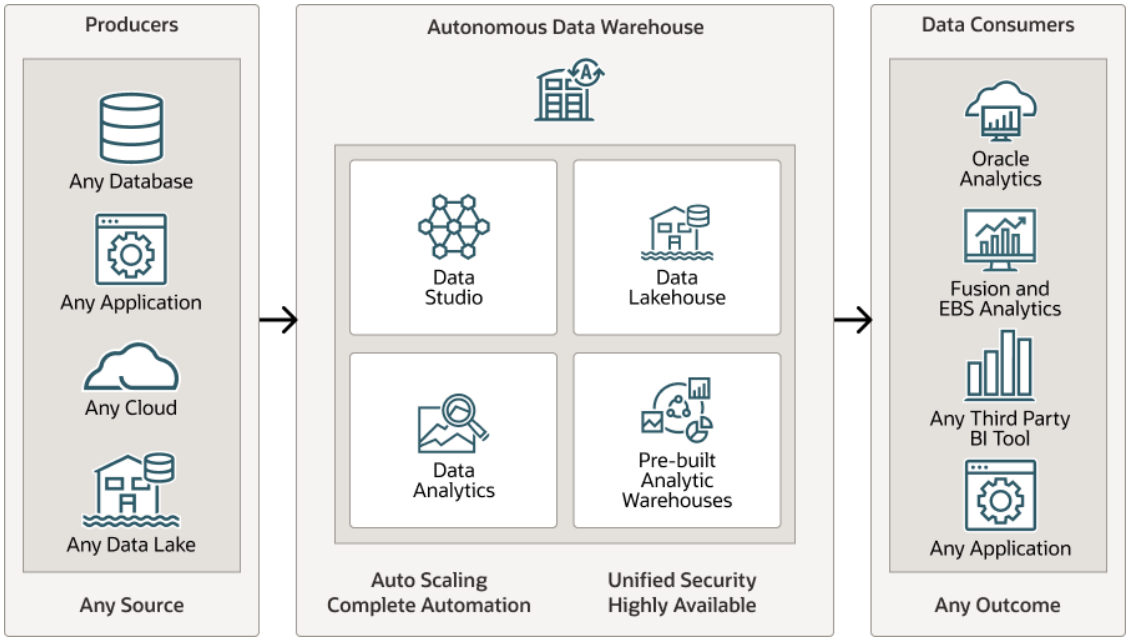
Data lakes are a key part of current data management architectures with data stored across object store offerings from Oracle, Amazon, Azure, Google, and other vendors.

Data lakes augment and complement data warehouses:

- As a data processing engine for ETL: This allows you to reduce the data warehousing workload.
- By storing data that may not be appropriate for a data warehouse: This includes log files, sensor data, IoT data, and so on. These source data tend to be voluminous with low information density. Storing this data in an object store might be more appropriate than in a data warehouse, while the information derived from the data is ideal for SQL analytics.
- For use with data science and business analytics: It is easy to upload files into the data lake and then use various processing methods over that data (Spark, Python, and so on).

  Business analysts using Autonomous Database can easily take advantage of these data sets without ETL. You can combine the data sets with data in your warehouse to gain new insights. For example, an analyst uploads third party customer demographic files to object storage and then immediately uses that data with data in the warehouse to perform customer segmentation analyses, blending the demographic data with existing customer and sales data.

Autonomous Database's deep integration with the data lake represents a new category in modern data management: the data lakehouse. Autonomous Database simplifies access to the data lake by providing rich and highly performant SQL access and embedded analytics, including: machine learning, graph, spatial, JSON, and more. This open access allows any SQL based business intelligence tool or application to benefit by using data stored in multiple places without needing to understand the complexities of the data lake.



Integrating Autonomous Database and the Data Lake

Autonomous Database supports integrating with data lakes not just on Oracle Cloud Infrastructure, but also on Amazon, Azure, Google, and more. You have the option of loading data into the database or querying the data directly in the source object store. Both approaches use the same tooling and APIs to access the data. Loading data into Autonomous Database will typically offer significant query performance gains when compared to querying object storage directly. However, querying the object store directly avoids the need to load data and allows for an agile approach to extending analytics to new data sources. Once those new sources are deemed to have proven value, you have the option to load the data into Autonomous Database.

Security Credentials for Accessing Data in Object Stores

Autonomous Database supports multiple cloud services and object stores, including Oracle Cloud Infrastructure, Azure, AWS, Google, and others. The first step in accessing these stores is to ensure that security policies are in place. For example, you must specify authorization rules to read and/or write files in a bucket on object storage. Each cloud has its own process for specifying role based access control.

Review the following to set up the proper policies for your object storage platform:

> **Note:**
>
> Security principals are not required for data that is stored publicly.

- Use Resource Principal to Access Oracle Cloud Infrastructure Resources
- Use Amazon Resource Names (ARNs) to Access AWS Resources
- Use Azure Service Principal with DBMS_CLOUD
- Use Google Service Account with DBMS_CLOUD

Autonomous Database uses a credential object to represent the identity to access a cloud service. See CREATE_CREDENTIAL Procedure for more information on security credentials that are used to connect to external sources.

This identity may be an Oracle Cloud Infrastructure user or a resource principal, an Amazon Web Service ARN, an Azure service principal, or a Google service account. The credential is included with each Autonomous Database issued service request and the user action is authorized by the cloud resource. Without proper authorization, the service request fails, typically with a "not found" error.

Review the following blog posts to learn more about using the different types of cloud service credentials:

- Accessing Oracle Cloud Infrastructure Resources from Your Autonomous Database using Resource Principal
- Autonomous Database Now Supports Accessing the Object Storage with OCI Native Authentication
- Using role Amazon Resource Names (ARNs) to securely access AWS Resources from your Autonomous Database
- How to Easily Access Azure Resources from Your Autonomous Database: Once the security policies are in place, you can use both the Database Tools and `DBMS_CLOUD` APIs to access those services.

After security policies are in place you can use either the Autonomous Database Tools or `DBMS_CLOUD` APIs to access services.

Use Data Studio to Load and Link Data

Data Studio is designed for the business user and provides features for data loading that efficiently loads data from the data source and data linking that creates external tables that are used to query data dynamically.

You can start by creating a pointer to a cloud storage location. A cloud storage location is a bucket in an object store. Next, use the Database Tools to drag and drop the data to map object storage sources to target tables.

When creating the mapping, the schemas (columns and data types) for the underlying object store objects are automatically derived to simplify the job definition. Under the covers, Data Studio uses the Autonomous Database `DBMS_CLOUD` APIs to link to and load data.

See The Data Load Page for more information.

Load Data

You can use `DBMS_CLOUD` data loading APIs as part of your ETL processing flows. `DBMS_CLOUD` APIs simplify the loading process by encapsulating the complexities of dealing with different data sources and producing highly efficient loading jobs. You specify the source properties (for example, data location, file type, and columns for unstructured files) and their mapping to a target table. The documentation describes the details of loading data from a variety of sources and describes debugging data loads, including the processing steps and bad records that were encountered.

Loading data involves the following:

- Creating credentials
- Loading different types of files, including JSON, data pump, delimited text, Parquet, Avro, and ORC.

- Replication using Oracle GoldenGate, and more.

Some of the key `DBMS_CLOUD` APIs for loading data are:

- COPY_DATA Procedure: Loads data from object storage sources. Specify the source and how that source maps to a target table. The API supports a variety of file types, including: delimited text, Parquet, Avro, and ORC. There are numerous features for logging the processing.
- COPY_COLLECTION Procedure: Loads JSON data from object storage into collections.
- SEND_REQUEST Function and Procedure: The most flexible approach, `DBMS_CLOUD.SEND_REQUEST` sends and processes REST requests to web service providers. Use `DBMS_CLOUD.SEND_REQUEST` in your own PL/SQL procedures to load data that is not available in an accessible file store (for example, weather data from government services).

You can also use data pipelines for continuous incremental data loading from your data lake (as data arrives on object store it is loaded to a database table). Data import pipelines leverage the same underlying loading capabilities provided by `DBMS_CLOUD`.

The `DBMS_CLOUD_PIPELINE` package offers a set of APIs to create, manage, and run a pipeline, including:

- CREATE_PIPELINE Procedure: Creates a new pipeline.
- SET_ATTRIBUTE Procedure: Allows you to specify pipeline properties, including: a description of the source data and how it maps to a target, the load interval, the job priority and more.
- START_PIPELINE Procedure: Starts a pipeline for continuous incremental data loading.

See Using Data Pipelines for Continuous Load and Export for more information.

Link Data

Autonomous Database external tables are used to link to data. An external table accesses the files in object store directly without loading the data. This is ideal for cases where you may want to quickly explore data that is in object storage to understand its value. Applications query external tables as they would any other table in Autonomous Database, the location of the source is transparent. This means that any SQL-based tool or application can query across the data warehouse and data lake, finding new insights that would have otherwise been difficult to achieve.

External tables support a variety of file types, including: delimited text, JSON, Parquet, Avro, and ORC. Review the documentation to learn more about external concepts, including the external table types, validating sources, and leveraging Oracle Cloud Infrastructure Data Catalog to automatically generate external tables.

Listed below are a few of the `DBMS_CLOUD` APIs for working with external tables:

- CREATE_EXTERNAL_TABLE: Create an external table over object storage data.
- CREATE_EXTERNAL_PART_TABLE: Create a partitioned external table over object storage data.

  Partitioned external tables deliver performance benefits at query time. Partition pruning eliminates the need to scan data files that are not required for processing. For example, you may have 12 partitions, one for each month in 2021. A query for March 2021 only scans the data for that one month, or 1/12 of the data, dramatically improving performance.

  The `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` procedure understands standard data lake partitioning patterns, where each partition's data is stored in its own folder. The procedure automatically creates partitions based on the underlying data organization. Use

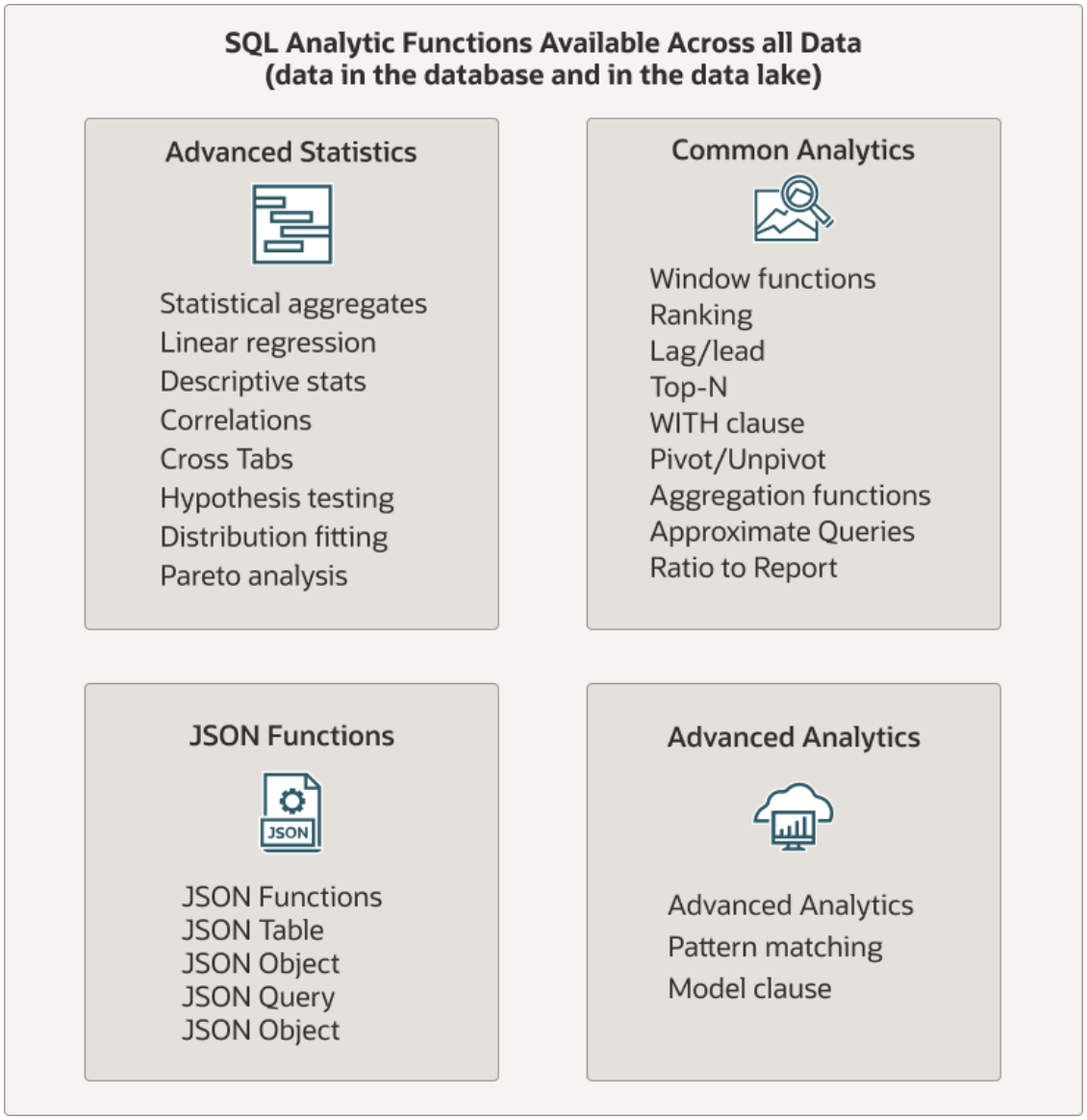
SYNC_EXTERNAL_PART_TABLE to update the table's partitions when the underlying data has changed.

- CREATE_HYBRID_PART_TABLE: Create a hybrid partitioned table.

  A hybrid partitioned table allows you to specify both partition data that should be stored in Autonomous Database and partition data to be stored externally. This allows you to keep "hot" data in the database for faster query performance and updates while archived data is stored in object store. Applications query the hybrid table without the need to understand where data resides.

Query Data Lakehouse Using SQL

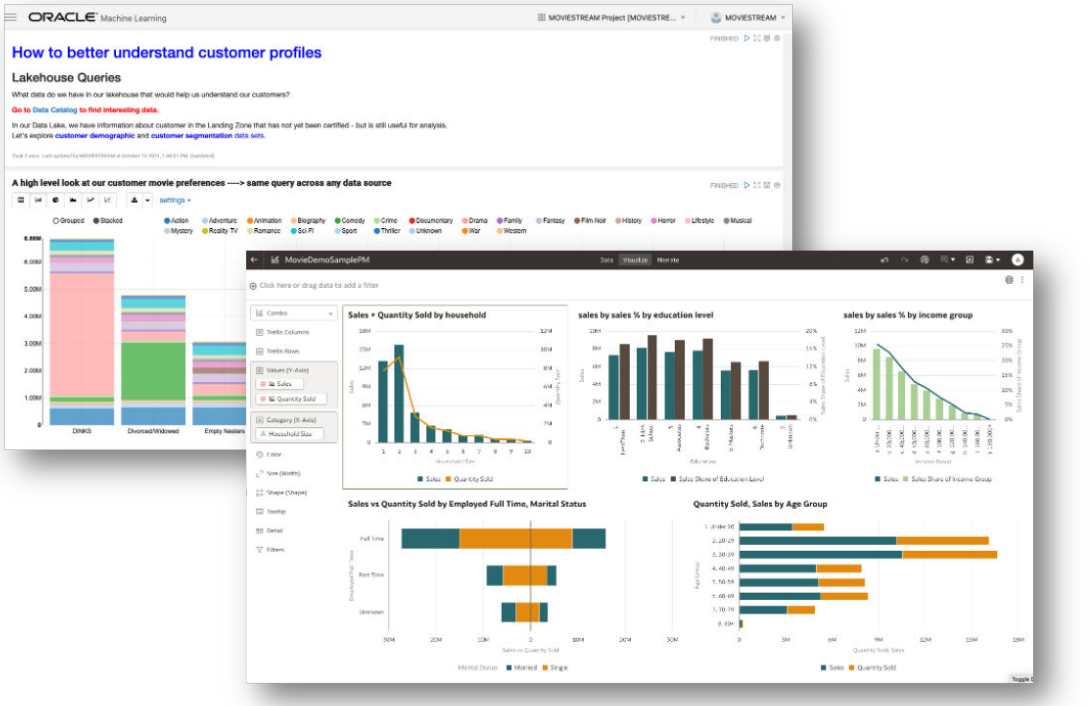
After you have integrated Autonomous Database with the data lake, you can use the full breadth of Oracle SQL for querying data across both the database and object storage.



The location of data is completely transparent to the application. The application simply connects to Autonomous Database and then uses all of the Oracle SQL query language to query across your data sets.

This allows you to:

- Correlate information from data lake and data warehouse.
- Access data from any SQL tool or application.
- Preserve your investment in tools and skill sets.
- Safeguard sensitive data using Oracle Database advanced security policies.



Advanced Analytics

Integrating the various types of data allows business analysts to apply Autonomous Database's built-in analytics across all data and you do not need to deploy specialized analytic engines.

Using Autonomous Database set up as a data lakehouse eliminates costly data replication, security challenges and administration overhead. Most importantly, it allows cross-domain analytics. For example, machine learning models can easily take advantage of spatial analytics to answer questions like, "What is the predicted impact of the movement of a hurricane on our product shipments?"

The following table provides a snapshot of the advanced analytic capabilities that you can take advantage of in Autonomous Database.

Advanced Analytics Type	Features
Graph Analytics	**Graph algorithms**: Library of 50+ built-in algorithms for detecting and evaluating communities, link prediction, path-finding, ranking and walking **Graph queries**: • Compute new metrics based on relationships between entities • Evaluate prediction results, discover new connections **Tooling for working with graphs**: • Graph modeling tool to map relational data to graphs • Browser-based notebooks for interactive analysis and collaboration • Integrated graph visualization See Oracle Graph for more information.
Spatial Analytics	**Geospatial data processing**: • Convert address data or place names to geospatial data • Prepare, validate and cleanse geospatial data **Geospatial data analysis**: Categorize or filter based on location and proximity **Map visualization**: Interactive analysis and reporting See Oracle Spatial for more information.
Machine Learning	**ML algorithms** • Fast model building and real time scoring • Choice of languages: SQL, R and Python **Collaborative tooling** • No-code AutoML automates model building, tuning and deployment • Notebooks offer interactive analyses and visualizations • Share notebooks and templates See Oracle Machine Learning for more information.
Analytic Views	**Shared business models**: • Define business hierarchies • Create metrics that leverage business hierarchies • Model shared by tools and applications (Oracle Analytics Cloud, Tableau, and so on) • SQL enhanced to leverage model **Tools for working with Analytic Views**: Data analysis tool simplifies creation of multidimensional model See Analytic Views for more information.

Collaborate Using Common Metadata

Historically, one of the major challenges in any data management solution is understanding the data itself.

You may have important questions about your data:

- What data is contained in a table?
- What is the meaning of a column?
- What is the reliability of the data? If it's not accurate enough for financial reporting, is it useful for marketing purposes?
- When was the data last updated?

- What is the schema for files contained in object storage?

Oracle Cloud Infrastructure offers a centralized data catalog, Oracle Cloud Infrastructure Data Catalog (Data Catalog), that provides the answers to these questions. Data Catalog allows you to quickly discover data across Object Storage, Autonomous Database, Oracle Databases, MySQL, Kafka, Hive, and more.

Automated metadata harvesting, for example from table and column definitions, derives technical metadata from these sources. It is also possible to derive metadata from unstructured sources in the object storage data lake. Once the sources have been harvested, business analysts and data stewards apply business terms and categories to the data. You now have a collaborative environment for data providers/consumers to search assets based on names, business terms, tags, and custom properties.

Autonomous Database integrates with Data Catalog, simplifying the administrative process and promoting semantic consistency across the lakehouse. The data lake metadata harvested by Data Catalog is automatically synchronized with Autonomous Database. This allows business analysts to immediately query data in the object store and combine that information with data in Autonomous Database using their favorite SQL based tool and application.

View details about how to set up the connection to Data Catalog

Spark on Autonomous Database

There is no single processing or analytic engine for object storage data. However, one of the leading processing engines is Apache Spark.

Spark is a distributed data processing framework that is used heavily in ETL and data science workloads. Spark is used by Oracle Cloud Infrastructure services, including:

- Oracle Cloud Infrastructure Data Flow: fully managed Apache Spark service.
- Oracle Cloud Infrastructure Data Integration: a fully managed cloud service that helps you with common extract, load, and transform (ETL) tasks such as ingesting data from different sources, cleansing, transforming, and reshaping that data, and then efficiently loading it to target data sources.
- Oracle Cloud Infrastructure Data Science: machine learning (ML) service that offers JupyterLab notebook environments and access to hundreds of popular open source tools and frameworks.Spark-based.

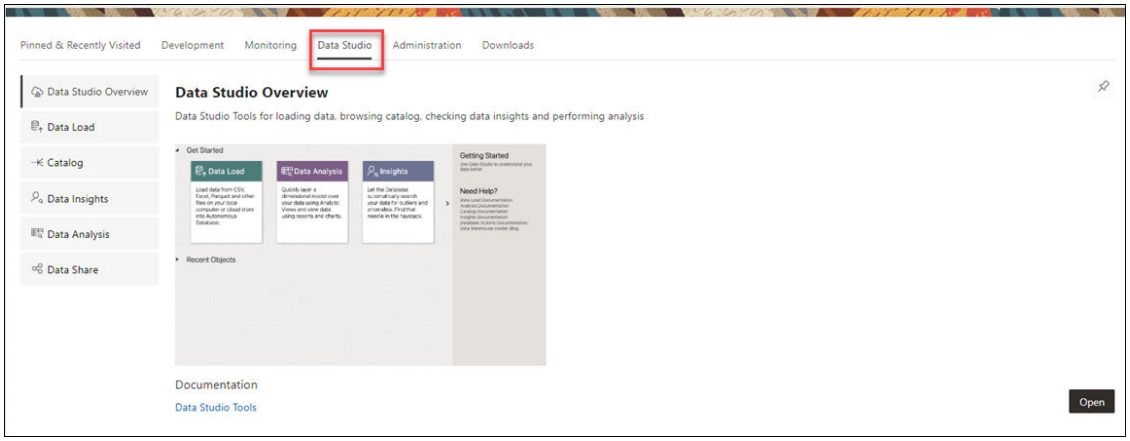
There is deep Spark integration with Autonomous Database. Spark on Oracle is an open source project that optimizes access to Autonomous Database from Spark-based applications. The Spark catalog is extended with the database data dictionary – think of the database as simply an additional namespace that is available. Queries that are executed thru Spark are automatically translated into highly efficient Oracle SQL by rewriting Spark against Autonomous Database. This means that on Autonomous Database the integration goes well beyond what most Spark 'connectors' provide (that is pushing projections and filters to the underlying system). Spark on Oracle is able to entirely push complex analysis pipelines containing all the analytical functions and operators of Spark SQL. For example, over 90% of TPC-DS queries were fully pushed down to Oracle Database for processing.

View the blog for Spark on Oracle and the open source project.

The Data Studio Overview Page

The Data Studio comprises of the Data Load, the Data Analysis, the Data Insights, Catalog and the Data Share tool.

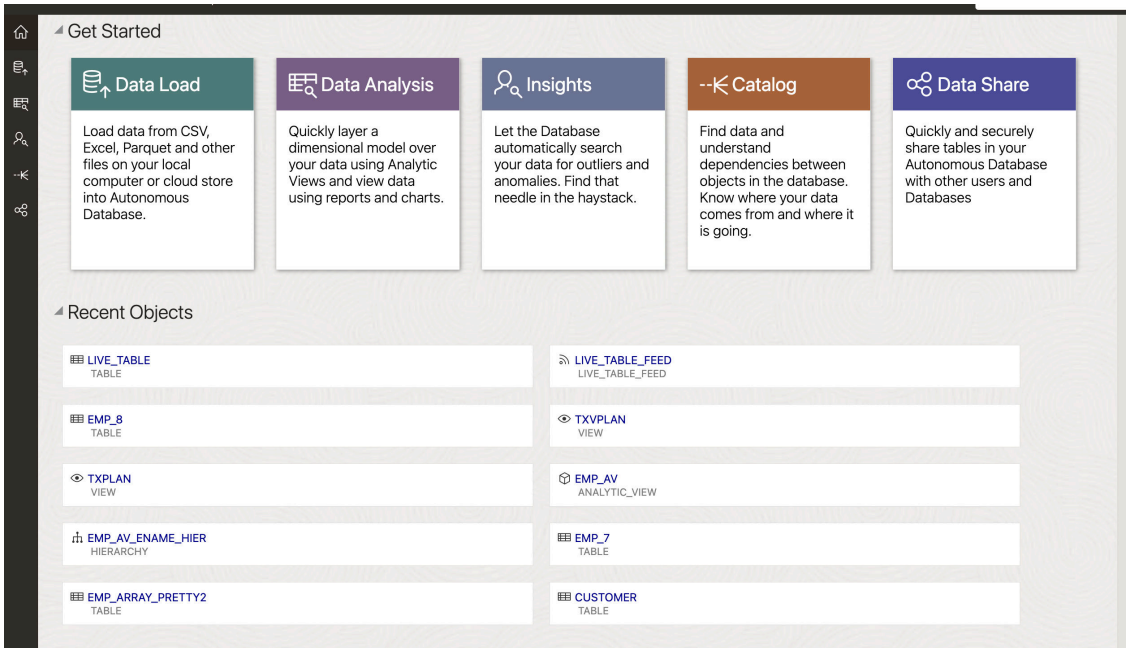
The Data Studio Tools enables you to load data from cloud and other diverse sources, analyzes it and gain insights from it. You can share the result of the analysis with other users. It is a one-stop application of your analytics tool from multiple data sources. This tool makes sure that there is seamless transition between different applications. The multiple ways of navigation do not impact the progress of your work. For instance, if you are working on data analysis and decide you need some additional data, you can navigate to the Data Load page, bring in the new contents, and return to your analysis in progress. Select Data Studio tab from the Launchpad. You can view the following page:



Data Studio Overview page

Click the **Data Studio Overview** menu from the **Data Studio** to view the Data Studio Overview page consists of four sections:

1. A navigation pane which consists of the following tools in the menu: Data Load, Data Analysis, Data Insights, the Catalog and the Data Share tools.
2. A widget with slides which displays information on the tools and their purpose. For example, the Data Load tool loads data from files on your computer or from cloud storage. Click **<** or **>** button to navigate back and forth the slides.
3. The right pane toolbar displays the help links to the Data Studio tool documentation.
4. A Recent Objects section which displays recently updated or created objects. Each entry has a context menu available at the end of the row. Click on the entries to view the details of the selected object.



Load your data from diverse sources such as files, databases and cloud storage, which are all consolidated in a single location. Some of the cloud data warehouses the Data Studio tool supports are Oracle Cloud Infrastructure (OCI), Amazon S3, Microsoft Azure Blob Storage and Google Cloud Storage.

Once your data is available, analyze it, generate insights, and create reports.

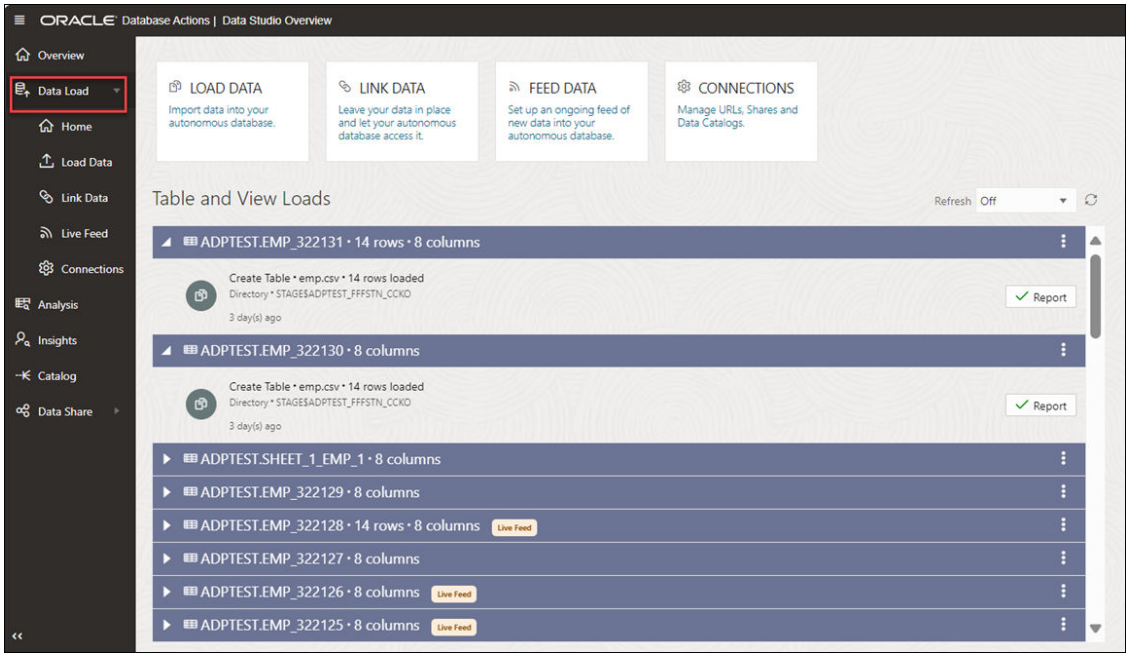
On the right of the Data Studio Overview page, refer to the Getting Started panel to know more about this tool.

> **Note:**
>
> You will lose your data once you click the Database Actions in the header or click the selector icon.

Let us run analysis with data from different sources and generate insights from it.

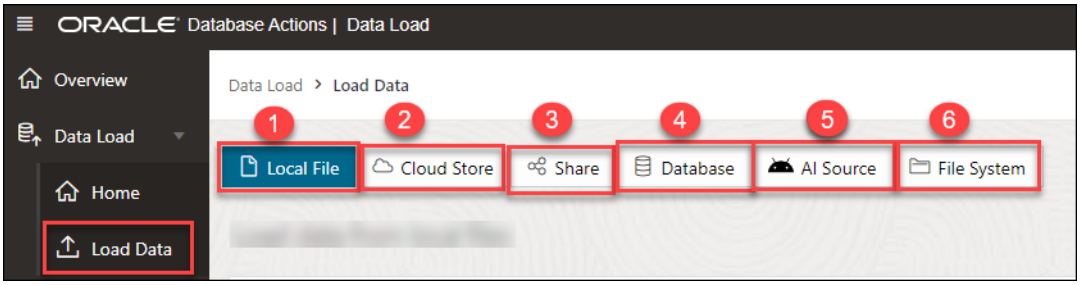
Load Data

Select the **Data Load** menu from the navigation pane in the Data Studio Overview page to load data from files on your local device, from remote databases, or from cloud storage buckets. Clicking Data Load opens Data Load Dashboard page.



The Data Load menu has the following submenus:

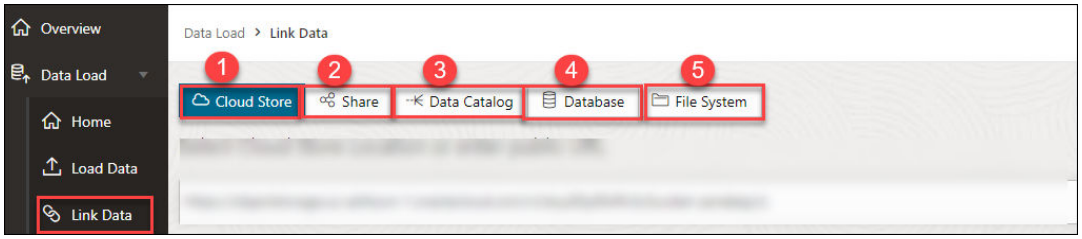
- **Home**: This menu takes you to the home page of the Data Load tool in the Launchpad. You can perform any of the following activities from the submenus such as Load local file, Load cloud store, Load Database Tables, Link Cloud Store, Link Database Tables, Create Live feed and search for cloud location and then select the Home menu. This does not break the continuity of the action you perform. Refer to The Data Load Page for more details.
- **Load Data**:
  Click **Load Data** to view the following options to load data:



  - **Load Local File**: This option enables us to load data from files on your local device. Refer to the Loading Data From Local Files section for more details.
  - **Load Cloud Store**: This menu enables you to load data from a cloud store location to a table in your Autonomous Database. Refer to the Loading Data from Cloud Storage section for more details.
  - **Load Share**: This menu enables you to load data from Shares where you can select tables from a Share. See Loading Data from a Share.
  - **Load Database**: This menu enables you to load data from tables in another database into your Oracle Autonomous Database. Refer to the Loading Data from other Databases section for more details.

  - **Load AI Source**: This menu enables you to load data from an AI source. See Loading Data from AI.
  - **Load File System**: This menu enables you to load data from file system directories to your Autonomous Database.

- **Link Data**:
  Click **Link Data** to view the following options of linking data:



  - **Link Cloud Store**: This menu enables you to link to data in cloud storage buckets. Refer to the Linking to Objects in Cloud Storage section for more details.
  - **Link Share**: This menu enables you to link a share with your Autonomous Database. See Linking to Share.
  - **Linking Data Catalog** : This menu enables you to link Data Catalog to your Autonomous Database. See Linking Data Catalog.
  - **Link Database**: This menu enables you to link to data in tables in another database from your Oracle Autonomous Database. Refer to the Linking to Other Databases section for more details.
  - **Linking File System**: This menu enables you to link file system directories to your Autonomous Database. See Linking to File System.

- **Live Feed**: This menu enables us to create a live table feed on demand, on a schedule, or as the result of a notification. Refer to the Feeding Data section for more details.
- **Connections**: You can view all the existing connections from this option. This option enables you to edit or delete a cloud storage link and manage catalogs and shares. You can also create new cloud storage links. Refer to theManaging Connections chapter for more details.

After you load the data using the data load job, it becomes available in the Data Load page. You can later inspect, review and delete the table on clicking the **Actions** icon. You could also reload the cart from the Reload Cart button to reload cart using the current cart in case you want to make changes to the table. This data will be the source data of the Analytic View. Use breadcrumbs at the top of the page to navigate back to the home page.

Follow the steps described in The Data Load Page chapter in this book to load data from Oracle Cloud Infrastructure to your Autonomous Database.

Data Analysis

Analytic views provide metadata that highlight the parts of your data that are most important to your organization. By creating an AV, you can improve the value of automatic insights and the analysis of your data because the system has this metadata.

Select the **Data Analysis** menu from the navigation pane in the Data Studio Overview page to create Analytic Views and analyze the data from the Analytic View. The table you loaded using the Load Data feature will be used to create the Analytic View. You can select both hierarchies and measures from Analytic Views. Analytic Views utilize diagrams, symbols, and text, that

represents how the data flows and the connection in between them. The Data Analysis tool helps you find the structural errors in the Analytic View you create. It spots anything from null values in a join column to duplicated fields. You can develop a visual representation of your analysis in the form of tables, Pivot tables and bar charts. You can also export the Analytic View to Tableau or Power BI for better visualization.

Create Analytic Views by referring to The Data Analysis Tool chapter in this book.

The **Related Insights** panel in the right works with the measures associated with the Analytic View to visualize data in the form of bar charts.

Automatic Insights

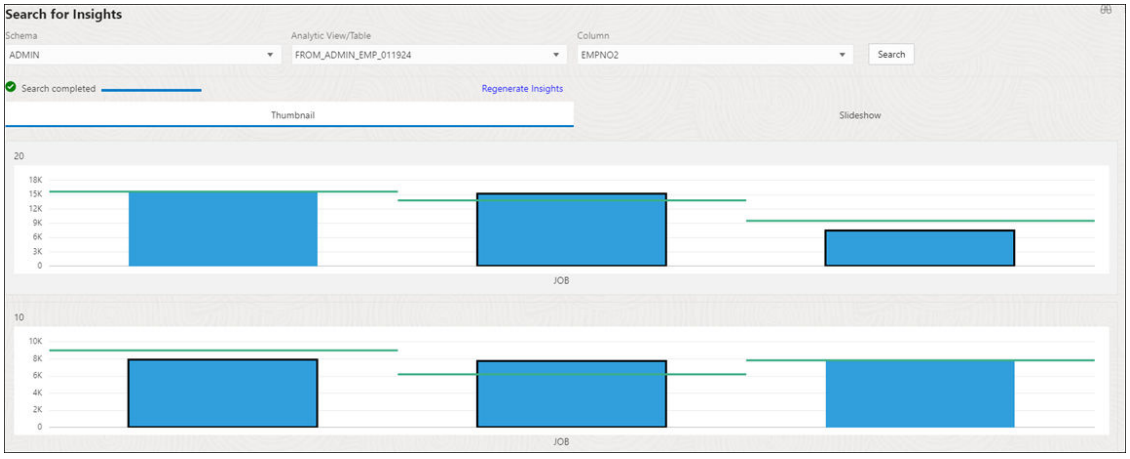
Select the **Automatic Insights menu** from the navigation pane in the Data Studio Overview page to generate insights automatically. The insights appear in the Data Insights dashboard as a series of bar charts.

By selecting the user, Analytic View, and the column name, you can view the data points of the actual values that deviate considerably from the forecast values. This type of predictive analytics will enable you to understand a pattern better and make better decisions. Then you can act on your data to optimize it or to improve its performance.

For example, consider that the fact table for the insights records values about different companies, and the measures of the fact table are Value (USD), Acquisition date and Acquisition year. Then you can generate insights for the Value measure. The dashboard will have a series of charts labeled Acquisition date and Acquisition year. Let's say you want to view the value of your company **A**. Click the chart whose sale you want to display for a particular range of Acquisition year. Clicking a chart displays more details about it.

Generate automatic insights by referring to The Data Insights Page in this book.

Below is the chart that displays the values and insights.



According to the insights generated by the tool, you notice that your company is lagging from the expected value by a significant number. In order to lead you need to change a few factors that affect the sales of your organization.

The insights that Data Insights generates for the analytic view are more useful than those for a table because of the additional metadata that an analytic view provides.

Catalog

Select the **Catalog** menu from the navigation pane in the Data Studio Overview page to view information about the upstream dependencies of the entity, how the entity was created and how it is linked to other entities.

This page lists details about the database objects such as cloud storage links and tables you create from the Data Analysis tool and the Data load tool. It also displays lists of the database dictionary objects such as tables, columns, and Analytic Views.

Use the Catalog page to locate objects in the catalog and perform tasks specific to those objects.

Refer to The Catalog Page chapter of this book for more information on the Catalog page.

Data Share

Select the **Data Share** menu from the navigation pane in the Data Studio Overview page to provide data shares and consume data shares.

See The Data Share Tool

High Availability

You can configure and manage Autonomous Data Guard and maintain continuous availability using disaster recovery.

- Use Application Continuity on Autonomous Database
  Autonomous Database provides application continuity features for making connections to the database.
- Use Standby Databases with Autonomous Data Guard for Disaster Recovery
- Backup and Restore Autonomous Database Instances
  This section describes backup and recovery tasks on Autonomous Database.
- Use OraMTS Recovery Feature on Autonomous Database
  Use Oracle MTS (OraMTS) Recovery Service to resolve an in-doubt Microsoft Transaction Server transaction.
- Use Backup-Based Disaster Recovery
  Backup-Based Disaster Recovery provides a lower-cost disaster recovery option for Autonomous Database (the RTO is higher with this option, as compared to using Autonomous Data Guard).
- Track Table Changes with Flashback Time Travel
  Use Flashback Time Travel to view past states of database objects or to return database objects to a previous state without using point-in-time media recovery.

Use Application Continuity on Autonomous Database

Autonomous Database provides application continuity features for making connections to the database.

- About Application Continuity on Autonomous Database
  Application Continuity masks outages from end users and applications by recovering the in-flight work for impacted database sessions following outages. Application Continuity performs this recovery beneath the application so that the outage appears to the application as a slightly delayed execution.
- Configure Application Continuity on Autonomous Database
  To configure Application Continuity you must enable application continuity for the database service your application uses and configure the failover type and the drain timeout. In addition, you must set several connection string parameters that enable high availability.

- Client Configuration for Continuous Availability on Autonomous Database
  You do not need to restart applications for planned maintenance activities when you enable Application Continuity and you follow the coding best practices.
- Notes for Application Continuity on Autonomous Database
  There are restrictions for Application Continuity on Autonomous Database, as follows:

About Application Continuity on Autonomous Database

Application Continuity masks outages from end users and applications by recovering the in-flight work for impacted database sessions following outages. Application Continuity performs this recovery beneath the application so that the outage appears to the application as a slightly delayed execution.

Your applications achieve *continuous availability* when planned maintenance, unplanned outages, and load rebalances of the database are hidden from the application. The combination of application coding best practices, application continuity configuration, and Autonomous Database ensures that your applications are continuously available.

The best approach for hiding planned maintenance activities from your applications is to transparently drain or failover applications. Oracle's connection pools and mid-tiers, including the WebLogic Server, Oracle Universal Connection Pool (UCP), OCI session pool and ODP.NET Unmanaged Provider are Fast Application Notification (FAN) aware and therefore are notified when maintenance is underway on Autonomous Database to allow graceful draining of work before maintenance. Application Continuity runs during planned maintenance to failover those sessions that do not drain in the predefined drain interval (5 minutes on Autonomous Database).

In order to hide unplanned outages resulting from a component or communication failure Oracle provides:

- **Notification.** FAN is the first step to hiding outages. FAN notifies clients and breaks them out of their current network wait when an outage occurs. This avoids stalling applications for long network waits. For Autonomous Database, FAN is handled at the driver and by the Autonomous Database cloud connection manger.

  FAN notification automatically triggers closing idle connections, opening new connections in the new service location, and allows a configurable time for active work to complete in the soon-to-be-shutdown service location. The major third-party JDBC mid-tiers, such as IBM WebSphere, allow for the same behavior when configured with UCP. For JDBC-based applications that cannot use UCP, Oracle provides solutions using Oracle Drivers and connection tests. On Autonomous Database FAN for planned maintenance is sent in-Band.
- **Recovery.** After the client is notified, failover handling with Transparent Application Continuity (TAC) or Application Continuity (AC) re-establishes a connection to the Autonomous Database and replays in-flight, uncommitted, work when possible. By replaying in-flight work, the application can usually continue executing without knowing that any failure happened.

You enable Application Continuity on Autonomous Database in one of two configurations, depending on the application:

- **Application Continuity (AC)**

  Application Continuity hides outages for thin Java-based applications, and Oracle Database Oracle Call Interface and ODP.NET based applications with support for open-source drivers, such as Node.js, and Python. Application Continuity rebuilds the session by recovering the session from a known point which includes session states and transactional states. Application Continuity rebuilds all in-flight work. The application continues as it was, seeing a slightly delayed execution time when a failover occurs.

- **Transparent Application Continuity (TAC)**

  Transparent Application Continuity (TAC) transparently tracks and records session and transactional state so the database session can be recovered following recoverable outages. This is done with no reliance on application knowledge or application code changes, allowing Transparent Application Continuity to be enabled for your applications. Application transparency and failover are achieved by consuming the state-tracking information that captures and categorizes the session state usage as the application issues user calls.

See Overview of Application Continuity for more information on Application Continuity.

> **Note:**
>
> By default Application Continuity is disabled on Autonomous Database.

Configure Application Continuity on Autonomous Database

To configure Application Continuity you must enable application continuity for the database service your application uses and configure the failover type and the drain timeout. In addition, you must set several connection string parameters that enable high availability.

- Configure Your Service to Enable Application Continuity
  Use `DBMS_APP_CONT_ADMIN` to enable Application Continuity or Transparent Application Continuity:
- Use Fast Application Notification (FAN)
  When connecting to Autonomous Database, the Oracle database auto-configures FAN. For application deployments with Autonomous Database, Fast Application Notification (FAN) events for unplanned outages are directed to the connection manager (CMAN) and no client application configuration steps required to use FAN.
- Configure Connection String for High Availability
  To maintain high availability, Oracle recommends that you set certain connection string parameters when you connect to Oracle Autonomous Database.
- Configure Driver Specific Client Options
  Depending on your client and your driver, you need to assure that the client is properly configured to use Application Continuity when you connect to Autonomous Database

Configure Your Service to Enable Application Continuity

Use `DBMS_APP_CONT_ADMIN` to enable Application Continuity or Transparent Application Continuity:

- **Application Continuity (AC)**: Set this failover option using the procedure `DBMS_APP_CONT_ADMIN.ENABLE_AC`. The `ENABLE_AC` procedure takes three parameters: `SERVICE NAME` is the service name to change, `FAILOVER_RESTORE`, set to `LEVEL1` to select Application Continuity(AC), and `REPLAY_INITIATION_TIMEOUT`, is the replay timeout that specifies how many seconds after a request is submitted to allow that request to replay.

  For example, as the ADMIN user, to enable Application Continuity for the `TPURGENT` service:

```
execute DBMS_APP_CONT_ADMIN.ENABLE_AC(
        'databaseid_tpurgent.adb.oraclecloud.com', 'LEVEL1', 600);
```

- **Transparent Application Continuity (TAC)**: Set this failover option using the procedure `DBMS_APP_CONT_ADMIN.ENABLE_TAC`. The `ENABLE_TAC` procedure takes three parameters: `SERVICE NAME` is the service name to change, `FAILOVER_RESTORE`, set to `AUTO` to select Transparent Application Continuity (TAC), and `REPLAY_INITIATION_TIMEOUT` is the replay timeout that specifies how many seconds after a request is submitted to allow that request to replay.

  For example, as the ADMIN user, to enable Transparent Application Continuity for the `TP` service with the replay timeout set to 20 minutes:

  ```
  execute DBMS_APP_CONT_ADMIN.ENABLE_TAC(
          'databaseid_tp.adb.oraclecloud.com', 'AUTO', 1200);
  ```

- **Disabled**: Disable failover using the procedure `DBMS_APP_CONT_ADMIN.DISABLE_FAILOVER()`.

  For example, as the ADMIN user, to disable failover for the `TP` service:

  ```
  execute DBMS_APP_CONT_ADMIN.DISABLE_FAILOVER(
          'databaseid_tp.adb.oraclecloud.com');
  ```

Find the Service Name Parameter for Application Continuity

Depending on your workload type, use a command similar to the following to `SELECT` from `DBA_SERVICES` on your database and identify the service where you want to enable Application Continuity:

- **Data Warehouse**

```
SELECT name, failover_type FROM DBA_SERVICES;
NAME                                                     FAILOVER_TYPE
-------------------------------------------------------- -------------
nvt21_adb1_low.adb.oraclecloud.com
nvt21_adb1_high.adb.oraclecloud.com
nvt21_adb1_medium.adb.oraclecloud.com
```

- **Transaction Processing** or **JSON Database**

```
SELECT name, failover_type FROM DBA_SERVICES;
NAME                                                     FAILOVER_TYPE
-------------------------------------------------------- ----------------
nvt21_adb1_tp.adb.oraclecloud.com
nvt21_adb1_tpurgent.adb.oraclecloud.com
nvt21_adb1_low.adb.oraclecloud.com
nvt21_adb1_high.adb.oraclecloud.com
nvt21_adb1_medium.adb.oraclecloud.com
```

Notice the `FAILOVER_TYPE` for the `high` service has the no value and indicates that Application Continuity is disabled.

Verify Application Continuity is Enabled for a Service

Depending on your workload type, check the output of the query on `DBA_SERVICES` to verify that Application Continuity is enabled.

- **Data Warehouse**

```
SELECT name, failover_type FROM DBA_SERVICES;
NAME                                                  FAILOVER_TYPE
----------------------------------------------------- -------------
nvt21_adb1_low.adb.oraclecloud.com
nvt21_adb1_high.adb.oraclecloud.com                   AUTO
nvt21_adb1_medium.adb.oraclecloud.com
```

- **Transaction Processing** or **JSON Database**

```
SELECT name, failover_type FROM DBA_SERVICES;
NAME                                                  FAILOVER_TYPE
----------------------------------------------------- -------------
nvt21_adb1_tp.adb.oraclecloud.com
nvt21_adb1_tpurgent.adb.oraclecloud.com               TRANSACTION
nvt21_adb1_low.adb.oraclecloud.com
nvt21_adb1_high.adb.oraclecloud.com                   AUTO
nvt21_adb1_medium.adb.oraclecloud.com
```

The `FAILOVER_TYPE` value for the `high` service is now `AUTO`, indicating that Transparent Application Continuity (TAC) is enabled and the `FAILOVER_TYPE` value for the `tpurgent` service is now `TRANSACTION`, indicating that Application Continuity (AC) is enabled.

Use Fast Application Notification (FAN)

When connecting to Autonomous Database, the Oracle database auto-configures FAN. For application deployments with Autonomous Database, Fast Application Notification (FAN) events for unplanned outages are directed to the connection manager (CMAN) and no client application configuration steps required to use FAN.

FAN is automatically handled for you by the client driver and by the Autonomous Database Connection Manager (CMAN):

- For planned maintenance events, FAN is sent in-band, directly to the drivers. This requires that applications use Oracle Pools or TAC for request boundaries, or use connection tests.
- The Oracle Database and Oracle client drivers drain on connection tests and at request boundaries.

See Client Configuration for Continuous Availability on Autonomous Database for more information.

Configure Connection String for High Availability

To maintain high availability, Oracle recommends that you set certain connection string parameters when you connect to Oracle Autonomous Database.

Set the `CONNECT_TIMEOUT`, `RETRY_DELAY`, `RETRY_COUNT`, and `TRANSPORT_CONNECT_TIMEOUT` parameters in the connection string when you connect to Oracle Autonomous Database. The connect strings embedded in the Oracle-supplied `tnsnames.ora` file are preconfigured with appropriate values for most applications. In some cases, depending on your applications needs, you may need to change the preconfigured values for a connection string.

Use this TNS for all Oracle clients version 12.2 or higher:

```
alias =
(DESCRIPTION =
```

```
(CONNECT_TIMEOUT= 90)(RETRY_COUNT=50)
(RETRY_DELAY=3)(TRANSPORT_CONNECT_TIMEOUT=3)
(ADDRESS_LIST =
  (LOAD_BALANCE=on)
  (ADDRESS = (PROTOCOL = TCP)(HOST=scan-host)(PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME = service-name)))
```

Use the following for JDBC connections using Oracle driver version 12.1 or earlier:

```
alias =
(DESCRIPTION =
(CONNECT_TIMEOUT= 15)(RETRY_COUNT=50)
(RETRY_DELAY=3)
(ADDRESS_LIST =
   (LOAD_BALANCE=on)
   (ADDRESS = (PROTOCOL = TCP)(HOST=scan-host)(PORT=1521)))
 (CONNECT_DATA=(SERVICE_NAME = service-name)))
```

Notes for connection strings:

- For JDBC and ODP clients, the pool connection wait time should be configured to be longer than the `CONNECT_TIMEOUT` in the connect string.
- Do not use Easy Connect Naming on the client because such connections do not have high-availability capabilities.

See Download Client Credentials (Wallets) for information on the `tnsnames.ora` file.

Configure Driver Specific Client Options

Depending on your client and your driver, you need to assure that the client is properly configured to use Application Continuity when you connect to Autonomous Database

- Configure JDBC Thin Driver
  Shows details to use Application Continuity with Autonomous Database with a client using the JDBC Thin Driver.
- Configure Oracle Call Interface (OCI) Driver
  Shows details to use Application Continuity with Autonomous Database with a client using the Oracle Call Interface (OCI) driver.
- Configure ODP.NET Unmanaged Provider Driver
  Shows details to use Application Continuity with Autonomous Database with a client using the ODP.NET Unmanaged Provider Driver.

Configure JDBC Thin Driver

Shows details to use Application Continuity with Autonomous Database with a client using the JDBC Thin Driver.

If your application uses the JDBC Thin Driver, follow these recommended practices:

1. Use JDBC Statement Cache for Coverage and Performance.

   For best coverage and performance, use the JDBC driver statement cache in place of an application server statement cache. This allows the driver to know that statements are closed and memory is to be freed at the end of requests.

   To use the JDBC statement cache, use connection property `oracle.jdbc.implicitStatementCacheSize` (`OracleConnection.CONNECTION_PROPERTY_IMPLICIT_STATEMENT_CACHE_SIZE`). The

statement cache is per connection. The value for the cache size matches your number of `open_cursors`. For example:

`oracle.jdbc.implicitStatementCacheSize=nnn` where *nnn* is typically between 10 and 100 and is equal to the number of open cursors your application maintains.

2. Tune the Garbage Collector.

   For many applications the default Garbage Collector tuning is sufficient. For applications that return and keep large amounts of data you can use higher values, such as 2GB or larger. For example:

   ```
   java -Xms3072m -Xmx3072m
   ```

   It is recommended to set the memory allocation for the initial Java heap size (ms) and maximum heap size (mx) to the same value. This prevents using system resources on growing and shrinking the memory heap.

3. When using the Universal Connection Pool (UCP), disable Fast Connection Failover. For example:

   ```
   PoolDataSource.setFastConnectionFailoverEnabled(false)
   ```

Configure Oracle Call Interface (OCI) Driver

Shows details to use Application Continuity with Autonomous Database with a client using the Oracle Call Interface (OCI) driver.

If the client application uses the Oracle Call Interface (OCI) Driver, follow this recommended practice:

- Replace `OCIStmtPrepare` with `OCIStmtPrepare2`. `OCIStmtPrepare()` has been deprecated since 12.2. All applications should use `OCIStmtPrepare2()`. Transparent Application Continuity (TAC) and Application Continuity (AC) allow `OCIStmtPrepare` but do not replay this statement.

Do not configure ONS servers in `oraaccess.xml`:

```
<ons>
   <servers>
   <!—Do not enter any values -->
   </servers>
</ons>
```

Also, for Autonomous Database Serverless, do not configure the <fan> section:

```
<fan>
<!-- only possible values are "trace" or "error" -->
    <subscription_failure_action>
   </subscription_failure_action>
</fan>
```

Configure ODP.NET Unmanaged Provider Driver

Shows details to use Application Continuity with Autonomous Database with a client using the ODP.NET Unmanaged Provider Driver.

The ODP.NET Unmanaged Provider driver automatically uses Application Continuity when Application Continuity is enabled on the database service that your application uses to connect to Autonomous Database.

When connecting an ODP.NET application to your Autonomous Database Serverless, do not configure ONS servers in `oraaccess.xml`:

```
<ons>
   <servers>
   <!—Do not enter any values -->
   </servers>
</ons>
```

Client Configuration for Continuous Availability on Autonomous Database

You do not need to restart applications for planned maintenance activities when you enable Application Continuity and you follow the coding best practices.

- Connect Using Database Services with Application Continuity Enabled
- Use Recommended Practices That Support Draining
  On Autonomous Database there is never a need to restart application servers when planned maintenance follows best practice.
- Steps for Using Application Continuity
  Perform these steps to use Application Continuity:
- Developer Best Practices for Continuous Availability
  Follow these best practices to code for continuous availability on Autonomous Database.

Connect Using Database Services with Application Continuity Enabled

Oracle database services provide transparency for the underlying Autonomous Database infrastructure.

The high availability and application continuity operations are predicated on the use of Autonomous Database connection services. To obtain application continuity, use a database service when you connect to your database.

The names of the predefined database services on Autonomous Database are different, depending on your workload, as described in Database Service Names for Autonomous Data Warehouse and Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database.

Use Recommended Practices That Support Draining

On Autonomous Database there is never a need to restart application servers when planned maintenance follows best practice.

For planned maintenance, the recommended approach is to provide time for current work to complete before maintenance is started. On Autonomous Database this happens automatically and work is drained before starting maintenance activities when you follow these guidelines:

- FAN with Oracle Connection Pools or Oracle Drivers
- Connection tests

Use draining in combination with your chosen failover solution for those requests that do not complete within the allocated time for draining. Your failover solution will try to recover sessions that did not drain in the allocated time.

Return Connections to the Connection Pool

The application should return the connection to the connection pool on each request. It is best practice that an application checks-out a connection only for the time that it needs it. Holding a connection instead of returning it to the pool does not perform. An application should therefore check-out a connection and then check-in that connection immediately the work is complete. The connections are then available for later use by other threads, or your thread when needed again. Returning connections to a connection pool is a general recommendation regardless of whether you use FAN to drain, or connection tests to drain.

Use an Oracle Connection Pool

Using a FAN-aware, Oracle connection pool is the recommended solution for hiding planned maintenance. As the maintenance progresses and completes, sessions are moved and rebalanced. There is no impact to users when your application uses an Oracle Pool with FAN and returns connections to the pool between requests. Supported Oracle Pools include UCP, WebLogic GridLink, Tuxedo, OCI Session Pool, and ODP.NET Managed and Unmanaged providers. No application changes whatsoever are needed to use FAN other than making sure that your connections are returned to pool between requests.

Use UCP with a Third-Party Connection Pool

If you are using a third party, Java-based application server, the most effective method to achieve draining and failover is to replace the pooled data source with UCP. This approach is supported by many application servers including Oracle WebLogic Server, IBM WebSphere, IBM Liberty, Apache Tomcat, Red Hat WildFly (JBoss), Spring, and Hibernate, and others.

Use Connection Tests

If you cannot use an Oracle Pool with FAN, then the Autonomous Database or provided client drivers will drain the session. When services are relocated or stopped during maintenance, or there is a switchover to a standby site using Autonomous Data Guard, the Oracle Database and Oracle client drivers look for safe places to release connections according to the following rules:

- Standard connection tests for connection validity at borrow or return from a connection pool
- Custom SQL tests for connection validity
- Request boundaries are in effect and the current request has ended
- Use Connection Tests with Autonomous Database
  You can add, delete, enable or disable connection tests for Autonomous Database.
- Use Connection Tests with Thin Java Driver
- Use Connection Tests with OCI Driver

Use Connection Tests with Autonomous Database

You can add, delete, enable or disable connection tests for Autonomous Database.

Use the view `DBA_CONNECTION_TESTS` to show the available connection tests.

For example:

```
SQL> EXECUTE
   dbms_app_cont_admin.add_sql_connection_test('SELECT COUNT(1) FROM DUAL');
SQL> EXECUTE
   dbms_app_cont_admin.enable_connection_test(dbms_app_cont_admin.sql_test,
                                              'SELECT COUNT(1) FROM DUAL');
SQL> SELECT * FROM DBA_CONNECTION_TESTS;
```

Configure the same connection test that is enabled in your database at your connection pool or application server. Also configure flushing and destroying the pool on connection test failure to at least two times the maximum pool size or `MAXINT`.

Use Connection Tests with Thin Java Driver

If you would like to use connection tests that are local to the driver and cannot use UCP's full FAN support:

- Enable `validate-on-borrow=true`
- Set the Java system properties:
  - `-Doracle.jdbc.fanEnabled=true`
  - `-Doracle.jdbc.defaultConnectionValidation=SOCKET`

And then use one of the following tests:

- `java.sql.Connection.isValid(int timeout)`
- `oracle.jdbc.OracleConnection.pingDatabase()`
- `oracle.jdbc.OracleConnection.pingDatabase(int timeout)`
- A `HINT` at the start of your test SQL:
  - `/*+ CLIENT_CONNECTION_VALIDATION */`

Use Connection Tests with OCI Driver

If you would like to use the OCI driver directly, use `OCI_ATTR_SERVER_STATUS`. This is the only method that is a code change. In your code, check the server handle when borrowing and returning connections to see if the session is disconnected. During maintenance, the value of `OCI_ATTR_SERVER_STATUS` is set to `OCI_SERVER_NOT_CONNECTED`. When using OCI session pool, this connection check is done for you.

The following code sample shows how to use `OCI_ATTR_SERVER_STATUS`:

```
ub4 serverStatus = 0OCIAttrGet((dvoid *)srvhp,
          OCI_HTYPE_SERVER,
        (dvoid *)&serverStatus, (ub4 *)0, OCI_ATTR_SERVER_STATUS,
      errhp);if (serverStatus ==
        OCI_SERVER_NORMAL)printf("Connection is
        up.\n");else if (serverStatus ==
        OCI_SERVER_NOT_CONNECTED) printf("Connection is down.\n");
```

Steps for Using Application Continuity

Perform these steps to use Application Continuity:

- As a prerequisite, enable and configure Application Continuity or Transparent Application Continuity (TAC) for your database service on Autonomous Database. See Configure Application Continuity on Autonomous Database for more information.
- Oracle strongly recommends that you use the latest client drivers. Oracle Database 19c client drivers and later provide full support for Application Continuity (AC) and for Transparent Application Continuity (TAC). Use one of the following supported clients drivers:
  - Oracle JDBC Replay Driver 19c or later. This is a JDBC driver feature provided with Oracle Database 19c for Application Continuity
  - Oracle Universal Connection Pool (UCP) 19c or later with Oracle JDBC Replay Driver 19c or later
  - Oracle Weblogic Server 12c with Active GridLink, or third-party JDBC application servers using UCP with Oracle JDBC Replay Driver 19c or later
  - Java connection pools or standalone Java applications using Oracle JDBC Replay Driver 19c or later
  - Oracle Call Interface Session Pool 19c or later.SQL*Plus 19c (19.8) or later
  - ODP.NET pooled, Unmanaged Driver 19c or later ("Pooling=true" default in 12.2 and later)
  - Oracle Call Interface based applications using 19c OCI driver or later

Return Connections to the Connection Pool

The application should return the connection to the Oracle connection pool on each request. Best practice for application usage is to check-out (borrow) connections for only the time that they are needed, and then check-in to the pool when complete for the current actions. This is important for best application performance at runtime, for rebalancing work at runtime and during maintenance and failover events. This practice is also important for draining.

When using an Oracle connection pool, such as Universal Connection Pool (UCP) or OCI Session Pool, or ODP.Net Unmanaged Provider or when using WebLogic Active GridLink, following this practice embeds request boundaries that Application Continuity uses to identify safe places to resume and end capture. This is required for Application Continuity and is recommended for Transparent Application Continuity.

Transparent Application Continuity, in addition, will discover request boundaries if a pool is not in use or when replay is disabled. The conditions for discovering a boundary are:

- No open transaction
- Cursors are returned to the statement cache or cancelled
- No un-restorable session state exists (refer to Clean Session State between Requests in this paper)

Enable Mutables Used in the Application

Mutable functions are functions that can return a new value each time they are executed. Support for keeping the original results of mutable functions is provided for `SYSDATE`, `SYSTIMESTAMP`, `SYS_GUID`, and sequence.`NEXTVAL`. If the original values are not kept and different values are returned to the application at replay, replay is rejected.

If you need mutables for PL/SQL, issue `GRANT KEEP` as required.

For example:

```
SQL> GRANT KEEP DATE TIME to adb_user;
SQL> GRANT KEEP SYSGUID to adb_user;
SQL> GRANT KEEP SEQUENCE mySequence to adb_user on mysequence.myobject;
```

Side Effects

When a database request includes an external call such as sending MAIL or transferring a file then this is termed a side effect.

Side effects are external actions, they do not roll back. When replay occurs, there is a choice as to whether side effects should be replayed. Many applications choose to repeat side effects such as journal entries and sending mail as duplicate executions cause no problem. For Application Continuity side effects are replayed unless the request or user call is explicitly disabled for replay. Conversely, as Transparent Application Continuity is on by default, TAC does not replay side effects. The capture is disabled, and re-enables at the next implicit boundary created by TAC.

Developer Best Practices for Continuous Availability

Follow these best practices to code for continuous availability on Autonomous Database.

Return Connections to the Connection Pool

The most important developer practice is to return connections to the connection pool at the end of each request. This is important for best application performance at runtime, for draining work and for rebalancing work at runtime and during maintenance, and for handing failover events. Some applications have a false idea that holding onto connections improves performance. Holding a connection neither performs nor scales.

Clean Session State between Requests

It is best practice to clean session state between database requests.

When an application returns a connection to the connection pool, cursors in FETCH status, and session state set on that session remain in place unless an action is taken to clear them. If your application is setting state, it is best practice to return your cursors to the statement cache and to clear application related session state to prevent leakage to later re-uses of that database session. Cleaning your session state ensures that TAC can discover boundaries.

To automatically clean your state between requests with Oracle Database 21c, set the service attribute `RESET_STATE=LEVEL1`. Doing this will avoid state leakage and fetching from cursors by later usage of the connection pool.

If you are using Oracle Database 19c, use `DBMS_SESSION.RESET_PACKAGE` to clear PL/SQL global variables, use `TRUNCATE` to clear temporary tables, `SYS_CONTEXT.CLEAR_CONTEXT` to clear context and cancel your cursors by returning them to the statement cache.

If your application is stateless, such as REST, APEX, Microservice, and most web applications, it is best practice to use `RESET_STATE`.

Do not embed COMMIT in PL/SQL and Avoid Commit on Success and Autocommit

It is recommended practice to use a top-level commit, (`OCOMMIT` or `COMMIT()` or `OCITransCommit`). If your application is using `COMMIT` embedded in PL/SQL or `AUTOCOMMIT` or

COMMIT ON SUCCESS, it may not be possible to recover following an outage or timeout. PL/SQL is not reentrant. Once a commit in PL/SQL has executed, that PL/SQL block cannot be resubmitted. Applications either need to unpick the commit which is not sound as that data may have been read, or for batch use a checkpoint and restart technique. When using AUTOCOMMIT or COMMIT ON SUCCESS, the output is lost.

If your application is using a top-level commit, then there is full support for Transparent Application Continuity (TAC), Application Continuity (AC), and TAF Select Plus. If your application is using COMMIT embedded in PLSQL or AUTOCOMMIT or COMMIT ON SUCCESS, it may not be possible to replay for cases where that the call including the COMMIT did not run to completion.

Use ORDER BY or GROUP BY in Queries

Application Continuity ensures that the application sees the same data at replay. If the same data cannot be restored, Application Continuity will not accept the replay. When a SELECT uses ORDER BY or GROUP BY order is preserved. In Autonomous Database the query optimizer most often uses the same access path, which can help in the same ordering of the results. Application Continuity also uses an AS OF clause to return the same query results where AS OF is allowed.

Considerations for SQL*Plus

SQL*Plus is often our go to tool for trying things out. SQL*Plus of course does not reflect our actual application that will be used in production, so it is always better to use the real application test suite to test your failover plan and to measure your protection. SQL*Plus is not a pooled application so does not have explicit request boundaries. Some applications do use SQL*Plus for example for reports. To use SQL*Plus with failover check the following:

1. FAN is always enabled for SQL*Plus. Use the recommended connect string that auto-configures ONS end points for you.
2. When using SQL*plus the key is to minimize round trips to the database: https://blogs.oracle.com/opal/sqlplus-12201-adds-new-performance-features
3. SQL*Plus is supported for TAC starting with Oracle Database 19c. For best results set a large arraysize. For example (set arraysize 1000). Avoid enabling serveroutput as this creates unrestorable session state.

Notes for Application Continuity on Autonomous Database

There are restrictions for Application Continuity on Autonomous Database, as follows:

- By default Application Continuity is disabled on Autonomous Database. See Configure Application Continuity on Autonomous Database for more information.
- For JDBC clients using Oracle Database JDBC driver and jars, draining for Java based applications on Autonomous Database requires that you use Oracle JDBC driver and Companion Jars 19.12.0.0 or newer, or apply a patch. See Patch Request 31112088 for more information.

Use Standby Databases with Autonomous Data Guard for Disaster Recovery

When you add an Autonomous Data Guard standby database, Autonomous Database provides data protection and disaster recovery for your database.

Autonomous Database Disaster Recovery Options

Autonomous Data Guard When you add an Autonomous Data Guard standby database, the system creates a standby database that continuously gets updated with the changes from the primary database. You can use Autonomous Data Guard with a standby in the current region, a local standby, or with a standby in a different region, a cross-region standby. You can also use Autonomous Data Guard with both a local standby and a cross-region standby.

Autonomous Data Guard is available for the following workload types:

- Data Warehouse
- Transaction Processing

Backup-Based Disaster Recovery uses backups to instantiate a peer database at the time of switchover or failover. This enables you to have a lower cost and higher Recovery Time Objective (RTO) disaster recovery option for your Autonomous Database, as compared with Autonomous Data Guard. For local backup-based disaster recovery, existing local backups are utilized. There are no additional costs for a local Backup-Based Disaster Recovery. Cross-Region Backup-Based Disaster Recovery incurs an additional cost. See Use Backup-Based Disaster Recovery for details on Backup-Based Disaster Recovery.

Backup-Based Disaster Recovery is available for all workload types.

- About Standby Databases
  Provides information about enabling and using Autonomous Data Guard for disaster recovery on Autonomous Database.
- Enable Autonomous Data Guard
- Add a Cross-Region Standby Database
  You can enable Autonomous Data Guard with a cross-region standby when Autonomous Database is available (Lifecycle State shows Available).
- Perform a Switchover
- Automatic Failover with a Standby Database
- Perform a Manual Failover
- Convert Cross Region Peer to Snapshot Standby
- Update Standby to Use a Backup Copy Peer
- Disable a Cross Region Standby Database
  Disabling a cross-region Autonomous Data Guard standby database terminates the standby database. If you later add a cross-region Autonomous Data Guard standby, the system creates a new cross-region standby database.
- Update Remote Peer Network ACLs
- Events and Notifications for a Standby Database
  You can use Oracle Cloud Infrastructure Events to be notified of and to specify rules to automatically respond to Autonomous Data Guard operations.

- Use the API
  Provides links for details on using API operations to manage Autonomous Data Guard.
- Autonomous Data Guard Notes
  Note the following for using Autonomous Database with an Autonomous Data Guard standby database:
- Autonomous Database Cross-Region Paired Regions

About Standby Databases

Provides information about enabling and using Autonomous Data Guard for disaster recovery on Autonomous Database.

When you use Autonomous Data Guard, the system creates a standby database that is continuously updated with the changes from the primary database. You can use Autonomous Data Guard with a standby in the current region, a local standby, or with a standby in a different region, a cross-region standby, or you can add both a local and a remote standby.

> **Note:**
>
> Autonomous Data Guard is available with the Data Warehouse and Transaction Processing workload types. Autonomous Data Guard is not available with the JSON and APEX workload types.

By selecting from the disaster recovery options that Autonomous Database provides, you can choose the features and options that meet your Recovery Time Objective (RTO) and Recovery Point Objective (RPO) requirements

By default, each Autonomous Database instance provides a local Backup-Based Disaster Recovery peer database.

To add automatic failover and to lower your Recovery Time Objective (RTO), you can use a local Autonomous Data Guard standby database.

To use the most resilient disaster recovery option that Autonomous Database offers, you can use a local Autonomous Data Guard standby database and a cross-region Autonomous Data Guard standby.

In addition, other options using Backup-Based Disaster Recovery enable you to provide lower cost and higher Recovery Time Objective (RTO) disaster recovery options, as compared to Autonomous Data Guard. See Use Backup-Based Disaster Recovery for details on Backup-Based Disaster Recovery.

- Autonomous Data Guard with Local Standby
  When you use an Autonomous Data Guard standby database in the current region, Autonomous Database provisions a local standby database and monitors the primary database; if the primary database goes down, the standby instance automatically assumes the role of the primary instance.
- Autonomous Data Guard with Cross-Region Standby
- Autonomous Data Guard Recovery Time Objective (RTO) and Recovery Point Objective (RPO)
- Autonomous Data Guard Operations
  Autonomous Database provides the following operations with Autonomous Data Guard:

- Autonomous Database Disaster Recovery Status
  Autonomous Database provides information about disaster recovery status on the Autonomous Database Details page.
- Autonomous Data Guard Events
  You can use Oracle Cloud Infrastructure events to respond when Autonomous Database changes its state due to an Autonomous Data Guard related event such as a failover or switchover operation.

Autonomous Data Guard with Local Standby

When you use an Autonomous Data Guard standby database in the current region, Autonomous Database provisions a local standby database and monitors the primary database; if the primary database goes down, the standby instance automatically assumes the role of the primary instance.

Local Autonomous Data Guard peer databases incur the additional cost of the base CPUs and the storage of the Primary database, including any auto-scaled storage usage, billed on the Primary database itself. Auto-scaled CPUs of the Primary database are not billed for additionally on the local Autonomous Data Guard peer database. The number of base CPUs is specified by the number of ECPUs (OCPUs if your database uses OCPUs) as shown in the **ECPU count** or **OCPU count** field on the Oracle Cloud Infrastructure Console.

With a local standby database, Autonomous Data Guard provides an identical standby database that allows the following, depending on the state of the primary database:

- If your primary database goes down, Autonomous Data Guard converts the standby database to the primary database with minimal interruption. After failover completes, Autonomous Data Guard creates a new standby database for you.
- You can perform a switchover operation, where the primary database becomes the standby database, and the standby database becomes the primary database.

Autonomous Database does not provide access to a standby database in the current region. You perform all operations, such as scaling up the **ECPU count** (**OCPU count** if your database uses OCPUs) and enabling compute auto scaling on the primary database, and Autonomous Data Guard then performs the same actions on the local standby database. Likewise, you only perform actions such as stopping or restarting the database on the primary database.

A local standby database is created in the same region as the primary database (current region). For better resilience, the standby database is provisioned as follows:

- In regions with more than one availability domain, a local standby database is provisioned automatically in a different availability domain than the primary database.
- In regions with a single availability domain, a local standby database is provisioned automatically in a different fault domain than the primary database (that is, on a different physical machine).

See Regions and Availability Domains for more information on availability domains.

All Autonomous Database features from the primary database are available when the local standby instance becomes the primary, after the system fails over or after you perform a switchover operation, including the following:

- **Database Options:** The **ECPU count** (**OCPU count** if your database uses OCPUs), Storage, Display Name, Database Name, Auto Scaling, Tags, and Licensing options have the same values after a failover to the standby database or after you perform a switchover.
- **OML Notebooks:** Notebooks and users created in the primary database are available in the standby.

- **APEX Data and Metadata:** APEX information created in the primary database is copied to the standby.
- **ACLs:** The Access Control List (ACL) of the primary database is duplicated for the standby.
- **Private Endpoint:** The private endpoint from the primary database applies to the standby.
- **APIs or Scripts:** Any APIs or scripts you use to manage the Autonomous Database continue to work without any changes after a failover operation or after you perform a switchover.
- **Client Application Connections:** Client applications do not need to change their connection strings to connect to the database after a failover to the standby database or after you perform a switchover.
- **Wallet Based Connections:** You can continue using your existing wallets to connect to the database after a failover to the standby database or after you perform a switchover.

Autonomous Data Guard with Cross-Region Standby

When you add a standby database in a different region, if the primary instance goes down, Autonomous Data Guard provides a standby database that is physically separated in a remote region. The standby database is available to assume the role of the unavailable primary instance.
A cross-region standby database is a replica of the primary database and may be used for recovery in case of failure or when the primary is not available. Enabling Autonomous Data Guard with a cross-region standby provides a low RTO solution for disaster recovery in the event an entire region is not available or when the primary database is down for any reason.

Autonomous Data Guard cross-region standby databases incur the additional cost of the base CPUs and twice the storage of the Primary database, including any auto-scaled storage usage, billed on the remote peer database. Auto-scaled CPUs of the Primary are not billed for additionally on the remote peer database. The number of base CPUs is specified by the number of ECPUs (OCPUs if your database uses OCPUs) as shown in the **ECPU count** (**OCPU count**) field on the Oracle Cloud Infrastructure Console.

Autonomous Data Guard allows you to create one remote standby database in a paired region. Paired regions are remote regions where you can create a cross-region standby database. See Autonomous Database Cross-Region Paired Regions for more information on paired regions.

You perform almost all operations, such as scaling up the **ECPU count** (**OCPU count** if your database uses OCPUs) and enabling compute auto scaling on the primary database. Autonomous Data Guard then performs the same actions on the cross-region standby database.

After you add a remote standby database, Autonomous Database provides access to the remote standby database from the Oracle Cloud Infrastructure Console. Autonomous Database provides access to the remote standby database so that you can perform some operations independently on the remote standby, such as configuring networks and VCNs for private endpoints and adding tagging to define keys and values that are not replicated between the primary database and the remote standby.

> **Note:**
>
> Autonomous Data Guard does not perform automatic failover for a cross-region standby. If the primary database is unavailable and a local standby is unavailable, you can perform a manual failover to make the standby database assume the primary role.

You cannot connect to a cross-region standby while the it operates as a standby database, and it is not available for read-only operations. You can connect to the database in the following cases:

- When the database assumes the primary role after a failover or switchover operation. See Perform a Switchoverand Perform a Manual Failover for more information.
- After you convert the standby database to a snapshot standby database. See Convert Cross Region Disaster Recovery Peer to a Snapshot Standby for more information.

The following areas have differences for failover or switchover from the primary database to the remote standby database, when compared to failover or switchover to a local standby:

- **Display Name**: The display name has a "_Remote" extension.
- **OML Notebooks:** After a cross-region switchover or failover, OML notebooks from primary that was switched over or failed over are not present in the primary database (the current primary database after the role change). New OML notebooks can be created.
- **Private Endpoint**: You can independently configure and update private endpoints on the standby database before failover or before you perform a switchover. This allows you to have a private endpoint that is configured differently, after failover or after you perform a switchover. Autonomous Database does not keep the networking configuration synchronized from the primary to the standby.

  VCN Peering and domain forwarding are required for wallets to work across regions, with Autonomous Databases with a private endpoint with an Autonomous Data Guard standby, where the primary and the standby database are in different VCNs. See Remote VCN Peering using an RPC and DNS in Your Virtual Cloud Network for information on VCN peering and domain forwarding.
- **Network Access Control List**: By default a disaster recovery primary database and remote peer databases use the same network Access Control Lists (ACLs). Optionally, you can independently edit network ACLs on a remote peer database. This allows you to use different ACLs on a remote peer database.

  See Update Remote Peer Network ACLs for more information.
- **Tags**: Tags are handled independently on a disaster recovery primary and on remote peer database. This means:
  - When you add, remove, or update a tag on a remote peer, the change applies only on the remote peer database.
  - When you add, update, or remove a tag on the primary, the tag is not added, updated, or removed on remote peer databases.
- **APIs or Scripts:** Any APIs or scripts you use to manage the Autonomous Database need to be updated to call the APIs on the primary database, the current primary database, after a failover or after you perform a switchover.

  For mTLS connections, you must download a wallet from the primary database, the current primary database, after a failover or after you perform a switchover. See Cross-Region Disaster Recovery Connection Strings and Wallets for more information.

- **Client Applications**: Client applications need to connect using the connection strings and wallet you download from the primary database, the current primary database, after a failover or after you perform a switchover. See Cross-Region Disaster Recovery Connection Strings and Wallets for more information.
- **Wallet Based Connections:** You must download a wallet and connect using the connection strings from the primary database, the current primary database, to connect to the database after a failover or after you perform a switchover. See Cross-Region Disaster Recovery Connection Strings and Wallets for more information.
- **Autonomous Database Tools**: The tools have different URLs in the primary database, the current primary database, after a failover or after you perform a switchover (the tools URLs do not change for a switchover or failover to a local standby):
  - Database Actions
  - Oracle APEX
  - Oracle REST Data Services (ORDS)
  - Graph Studio
  - Oracle Machine Learning Notebooks
  - Data Transforms
  - MongoDB API
- **Oracle Cloud Infrastructure Object Storage Usage**: After you failover or switchover from the primary database to the standby database, on the primary database (the current primary database) the credentials and the URLs that provide access to Object Storage continue to work as they did before the failover or the switchover, providing access to the following:
  - External tables
  - External partitioned tables
  - External hybrid partitioned tables

> **Note:**
>
> This applies when the Object Storage is available. For rare scenarios when the Object Storage is not available, Oracle recommends having Object Storage backups or replication to a different region. If the Object Storage is not available (that is the Object Storage resource you used with the Primary before a switchover or failover), you may update your user credentials and parameters that set URLs for Object Storage so that the parameters specify values to access an available region's Object Storage. See Using Replication for more information.

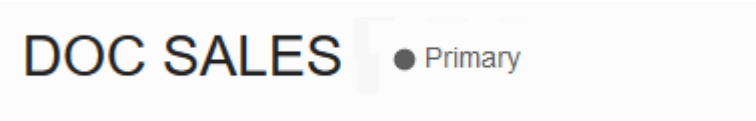
- Autonomous Data Guard Database Role
  After you add a cross-region standby database, each database has a designated role: primary, standby, or snapshot standby.
- Autonomous Data Guard Cross Region Failover and Switchover
  You can have one local disaster recovery peer and optionally you can add one cross-region peer. In both the local and cross-region cases, either peer can be a Backup-Based Disaster Recovery copy or an Autonomous Data Guard standby.
- Autonomous Data Guard Database Cross Region Backup and Restore
  After you add an Autonomous Data Guard cross-region standby database, backup and restore from backup is handled as follows:

- Cross-Region Disaster Recovery Connection Strings and Wallets
- Autonomous Data Guard with Customer Managed Keys
  When you add an Autonomous Data Guard cross-region standby, there are special considerations when the primary database is using customer-managed keys, or if you want to switch to using customer-managed keys on the primary database.
- Replicating Backups to a Cross-Region Autonomous Data Guard Standby
  When you add a cross-region Autonomous Data Guard standby you can enable cross-region backup replication so that automatic backups from the primary are also available on the remote region.

Autonomous Data Guard Database Role

After you add a cross-region standby database, each database has a designated role: primary, standby, or snapshot standby.

The role specifies the current state of a database, primary, standby, or snapshot standby, and this value changes after you perform a switchover or a failover or after you convert a standby database to a snapshot standby. You can view the Autonomous Database role in the icon that shows next to the display name on the Autonomous Database Information page. For example:

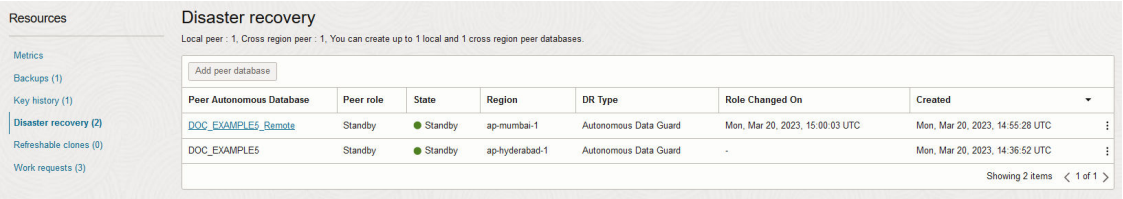


After you add a cross-region standby database, you can view the role in the **Disaster recovery** area on the details page. The role is one of:

- The **Role** shows **Primary** on the primary database.
- After a switchover or failover, the same database shows the **Role Standby**.
- After you convert a cross-region peer to a snapshot standby, the database shows the **Role Snapshot Standby**.

To view details for the peer, on the Autonomous Database Information page, under **Resources** select **Disaster recovery**:

- **Standby** (local): the **Peer role** column shows **Standby** and the database has the same display name in the **Peer Autonomous Database** column. The **Region** column shows the name of the current region.
- **Standby** (cross-region) the **Peer role** column shows **Standby** for a remote standby database and the database has the same with an "_Remote" extension in the **Peer Autonomous Database** column. You can click the link to access the remote database. The **Region** column shows the name of the remote region.
- **Snapshot standby** (local): the **Peer role** column shows **Snapshot standby** and the database has the same display name in the **Peer Autonomous Database** column. The **Region** column shows the name of the remote region.



Autonomous Data Guard Cross Region Failover and Switchover

You can have one local disaster recovery peer and optionally you can add one cross-region peer. In both the local and cross-region cases, either peer can be a Backup-Based Disaster Recovery copy or an Autonomous Data Guard standby.

With both a current region and a cross-region Autonomous Data Guard standby database, depending on the state of the primary database, you have the following options:

- If your primary database goes down and the local standby database is available, Autonomous Data Guard automatically performs failover to convert the local standby database to the primary database, with minimal interruption. After failover completes, Autonomous Data Guard creates a new local standby database for you. If automatic failover is not possible, you have the option to perform a manual failover.

  Autonomous Data Guard continues to use the same cross-region standby.
- If your primary database goes down and the local standby database is not available, you can perform a manual failover to the cross-region standby database and the cross-region standby database becomes the primary database.

  In this case, after the failover completes, Autonomous Data Guard does not create a new local standby database (by default you have a backup copy peer).
- You can perform a switchover operation, where the primary database becomes the local standby database, and the local standby database becomes the primary database.

  Autonomous Data Guard continues to use the same cross-region standby.
- You can perform a switchover operation, where the cross-region standby database becomes the primary database (and the database that was the primary is recreated as a new standby database so that it becomes the standby database).

  A switchover changes the roles of the primary and the standby database. If you perform a switchover two times, the primary database returns to again be the primary database.

Autonomous Data Guard Database Cross Region Backup and Restore

After you add an Autonomous Data Guard cross-region standby database, backup and restore from backup is handled as follows:

- If the primary database is restored from a backup, a new remote standby is created from the restored primary database.
- Automatic Backups are only taken on the primary database (the database showing **Role**: **Primary**). For example, after a switchover or failover, the database with the primary role starts to perform automatic backups. The database with the standby role no longer takes backups. If you switchover again, the database that becomes the primary role database starts taking backups again.
- You cannot restore or clone from a backup when either the primary database or the standby database is in the **Standby** role. Backups are only taken on the database in the **Primary** role, and the restore operation is not available from the Oracle Cloud Infrastructure Console on the **Standby** database.

Cross-Region Disaster Recovery Connection Strings and Wallets

When you add an Autonomous Data Guard cross-region (remote) standby database, or when you use a cross-region Backup-Based Disaster Recovery peer, the wallet and connection string from the primary database contains only the hostname of the primary database.

In addition, the wallet and connection string from the remote database contains only the hostname of the remote database. This applies for both instance and regional wallets.

Oracle recommends that you configure your applications running on the Primary Role database to use the wallet or connection string downloaded from the Primary database. For applications that run on the remote database, use the wallet or connection string downloaded from the remote database (where the remote database is the current primary database after a failover or after you perform a switchover). You can obtain these connection strings or the wallet by clicking **Database connection** on the Oracle Cloud Infrastructure Console.

For example, if your cross-region Autonomous Data Guard is setup with the primary in Ashburn (IAD) and a cross-region standby in Phoenix (PHX), Oracle recommends that your mid-tier applications running in IAD use the connection string or wallet from the primary database in IAD, and your corresponding applications that run in PHX after a failover or after you perform a switchover, use the connection string or wallet from the standby database in PHX. During regional failover or switchover, Oracle recommends failing over both your database and your mid-tier applications to the new Primary role database for optimum performance and to minimize any cross-regional latency.

See Download Client Credentials (Wallets) for more information.

If required by your application, you may manually construct connection strings containing both primary and remote database hostnames, to support connecting to either instance that is available and open for connections automatically, the primary or the remote database.

For details on the steps to manually create these connection strings see:

- Configure your mid-tier applications for cross-region disaster recovery
- Autonomous Data Guard Notes

Autonomous Data Guard with Customer Managed Keys

When you add an Autonomous Data Guard cross-region standby, there are special considerations when the primary database is using customer-managed keys, or if you want to switch to using customer-managed keys on the primary database.

In order for a remote standby to be able to use the same master encryption key as the primary database, the master encryption key must be replicated to the remote region. Replication of vaults and keys across regions is only possible if you select the **virtual private vault** option when you create the vault.

Consider the following cases:

- Adding an Autonomous Data Guard remote standby is allowed if the Autonomous Database is using customer-managed keys. When the database is using a customer-managed key and you add an Autonomous Data Guard cross-region standby, the **Region** list in the **Add peer database** dialog only shows the regions that contain the replicated vault and keys. If you don't see a remote region listed, you need to replicate your vault and keys to the region where you want your standby database (this must be a paired region).
- Switching to customer-managed keys is allowed on the primary when you have an Autonomous Data Guard cross-region standby. In the case when the database is using Oracle-managed keys and you switch to customer-managed keys on the primary, you only

see the keys that are replicated in both the primary and the standby regions. The Manage Encryption Key **Vault** and **Master encryption key** lists only show vaults and keys that are replicated across both the primary and the standby regions. If you don't see a key listed, replicate your vault and keys to a paired region.

See the following for more information:

- Replicating Vaults and Keys
- Overview of Vault
- Autonomous Database Cross-Region Paired Regions

Replicating Backups to a Cross-Region Autonomous Data Guard Standby

When you add a cross-region Autonomous Data Guard standby you can enable cross-region backup replication so that automatic backups from the primary are also available on the remote region.

By default the backups taken on the primary are not replicated to a cross-region standby database. When you enable cross-region backup replication, up to 7 days of automatic backups for the primary are replicated to a cross-region standby database. When this feature is enabled, automatic backups are available in the remote region as follows:

- After a switchover or failover you can restore or clone to any timestamp in the past seven (7) days, or to any timestamp in the specified retention period when the retention period is set to less than seven days.
- All backups for the primary that are replicated to the remote region are deleted on the remote region peer after seven days, or after the retention period number of days when the retention period is set to less than seven days.
- You cannot modify the backup retention period for replicated backups, except if you modify the backup retention period on the primary to specify a value less than seven days. In this case, the retention period for replicated backups on the remote region matches the automatic backup retention period set on the primary.

Cross-region backup replication incurs an additional cost. See Oracle Autonomous Database Serverless Features Billing for more information.

See Add a Cross-Region Standby Database and Enable or Disable Backup Replication for an Existing Cross Region Standby for more information.

Note the following for cross-region automatic backup replication:

- After a switchover or a failover, while the cross-region database is in the primary role, backups are taken on the current primary and are replicated to the current (remote) standby.
- In the remote region you can create a clone from a replicated backup while the database is in the standby role.

Autonomous Data Guard Recovery Time Objective (RTO) and Recovery Point Objective (RPO)

Autonomous Data Guard monitors the primary database and if the instance goes down, the local standby instance assumes the role of the primary instance, according to the Recovery Time Objective (RTO) and Recovery Point Objective (RPO).

If a local Autonomous Data Guard standby instance is not available and you have enabled cross-region disaster recovery, you can manually fail over to the cross-region standby.

If you do not add a cross-region Autonomous Data Guard standby, you have the option to add a cross-region Backup-Based Disaster Recovery peer. See Backup-Based Disaster Recovery Recovery Time Objective (RTO) and Recovery Point Objective (RPO) for details on the RTO and RPO with Backup-Based Disaster Recovery.

The RTO is the maximum amount of time required to restore database connectivity to a standby database after a manual or automatic failover is initiated. The RPO is the maximum duration of potential data loss on the primary database.

Local Autonomous Data Guard Standby

When you add a local standby database Autonomous Data Guard provides these options for failover or switchover:

- **Automatic Failover or Switchover**:

  When you enable Autonomous Data Guard you can select a data loss limit. The default data loss limit for automatic failover is 0 (valid values are 0 to 3600 seconds). For example, a data loss limit of 0 means that Autonomous Data Guard only performs automatic failover when there is no data loss. This means if Autonomous Data Guard can verify that there is no data loss, it automatically fails over in case of a problem. When there is a problem and Autonomous Data Guard determines that the possible data loss is greater than the data loss limit, automatic failover does not happen and you have the option to perform a manual failover.

- **Manual Failover**: the RTO is two (2) minutes and the RPO 10 seconds

Cross-Region Autonomous Data Guard Standby

When you add a cross-region standby database, the RTO and RPO numbers for failover to the Autonomous Data Guard cross-region standby are as follows:

- **Switchover**: the RTO is fifteen (15) minutes and RPO is zero (0).
- **Automatic Failover**: Not available
- **Manual Failover**: the RTO is fifteen (15) minutes and RPO is up to one (1) minute.

See the following for more information:

- Automatic Failover with a Standby Database for details on automatic failover.
- Perform a Switchover
- Perform a Manual Failover
- Service Level Objectives (SLOs)

Autonomous Data Guard Operations

Autonomous Database provides the following operations with Autonomous Data Guard:

- Enable: If you are using Backup-Based Disaster Recovery, you can update your disaster recovery type to local (current region) Autonomous Data Guard, or you can add an Autonomous Data Guard cross-region standby.

  See Enable Autonomous Data Guard and Add a Cross-Region Standby Database for details.

- Disable: If you have a local standby database or a cross-region standby database, you can change the disaster recovery type to Backup-Based Disaster Recovery for the local standby or you terminate the cross-region standby. In either case, disabling Autonomous Data Guard terminates the standby database.

See Update Standby to Use a Backup Copy Peer or Disable a Cross Region Standby Database for details.

- Switchover: When Autonomous Data Guard is enabled, switchover changes the roles of the primary and the standby, the standby database becomes the primary, and the primary database becomes the standby. If you have both a local standby database (current region), and a cross-region standby database (remote), you can choose to switchover to either the local standby or the remote standby.

  See Perform a Switchover for details.

- Manual Failover: If the primary database is not available you can perform a manual failover to change roles to make a standby database the primary database:

  – If a local standby is available, you can manually failover to the local standby (you do not have the option to failover to a remote standby if a local standby is available).

  – If a local standby is not available, you have the option to manually failover to a remote standby.

  See Perform a Manual Failover for details.

- Terminate: If you want to terminate the primary instance, select **More actions** and **Terminate**. Terminating the primary instance also terminates a local standby database.

  If you have both a local standby database (current region), and a cross-region standby database, you must terminate the cross-region standby database before you terminate the primary database.

  See Terminate a Cross-Region Standby Database for details.

Autonomous Database Disaster Recovery Status

Autonomous Database provides information about disaster recovery status on the Autonomous Database Details page.

In the **Disaster recovery** area:

The **Role** field shows the role of the current database, as follows:

- When you have either a local backup copy peer or a local Autonomous Data Guard standby, the Oracle Cloud Infrastructure Console shows the **Role** field value **Primary**. Autonomous Database does not provide access to a local standby database (or to a local backup copy peer).
- When using either a cross-region backup copy peer or a cross-region Autonomous Data Guard standby, the Oracle Cloud Infrastructure Console shows the **Role** field value **Primary** if you are viewing the primary database and shows **Standby** if you are viewing the details for the standby database.
- **Switchover**: Provides a link so that you can perform a switchover operation.
- **Failover**: When the primary database is not available and you have a local standby and automatic failover was not successful, the failover link allows you initiate a manual failover.

  When the primary database is not available and you have a cross-region standby and failover to a local standby is not possible, the failover link allows you initiate a manual failover to the remote standby database.

To view the peer Autonomous Database information, under **Resources** click **Disaster recovery**. This area lists the peer autonomous database information. The **State** column shows the state of a standby database, as follows:

- **Provisioning**

  - This state shows when you enable Autonomous Data Guard and indicates that a standby database is provisioning (until the standby database state changes to **Standby**).
  - This state shows after a failover to a local standby when a local standby database is being recreated.
  - This state shows if a restore from backup operation is performed on the primary database, the local standby is recreated and the **State** column shows Provisioning.
- **Standby**: Indicates that a standby is available and ready for either a switchover or a failover operation.

> **Note:**
>
> When the standby database is stopped, the standby state shows **Standby**. A standby database never shows the **Stopped** state.

- **Role Change in Progress**: Indicates a failover or switchover operation started.

Autonomous Data Guard Events

You can use Oracle Cloud Infrastructure events to respond when Autonomous Database changes its state due to an Autonomous Data Guard related event such as a failover or switchover operation.

Autonomous Database events include the following:

- Begin automatic failover
- End automatic failover
- Begin disable Autonomous Data Guard
- Begin enable Autonomous Data Guard
- Begin failover
- Begin switchover
- End disable Autonomous Data Guard
- End enable Autonomous Data Guard
- End failover with failover result of success or failure.
- End switchover with switchover result of success or failure.

Based on events you can perform actions or send notifications. See Events and Notifications for a Standby Database for more information on using events and producing notifications.

Enable Autonomous Data Guard

To enable Autonomous Data Guard you update the disaster recovery type to use a standby database.

By default and at no additional cost, Autonomous Database provides a local backup copy peer for each Autonomous Database instance. You enable Autonomous Data Guard by changing the disaster recovery type to use a standby database. Autonomous Data Guard provides a

lower Recovery Time Objective (RTO), compared to using a backup copy peer, and provides for automatic failover to a local standby when the primary database is not available.

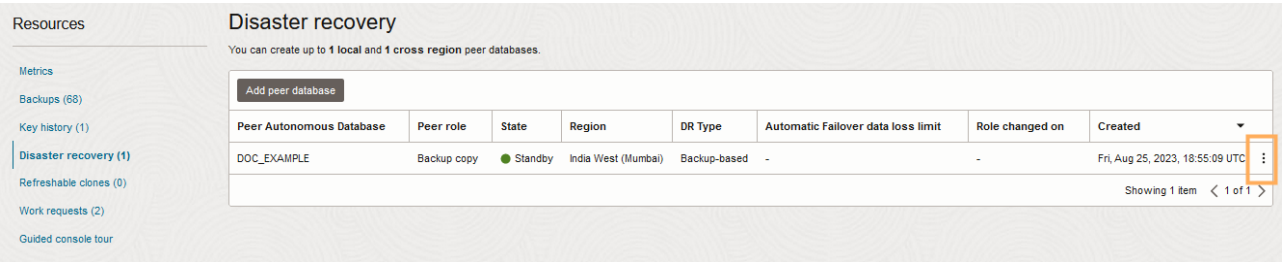
> **Note:**
>
> When you change your disaster recovery type to use a local standby database, you also have the option to add a second cross-region disaster recovery option, either a cross-region Autonomous Data Guard standby database or a cross-region backup copy peer.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload, click one of: Autonomous Data Warehouse or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To change your disaster recovery type to add a local Autonomous Data Guard standby database:

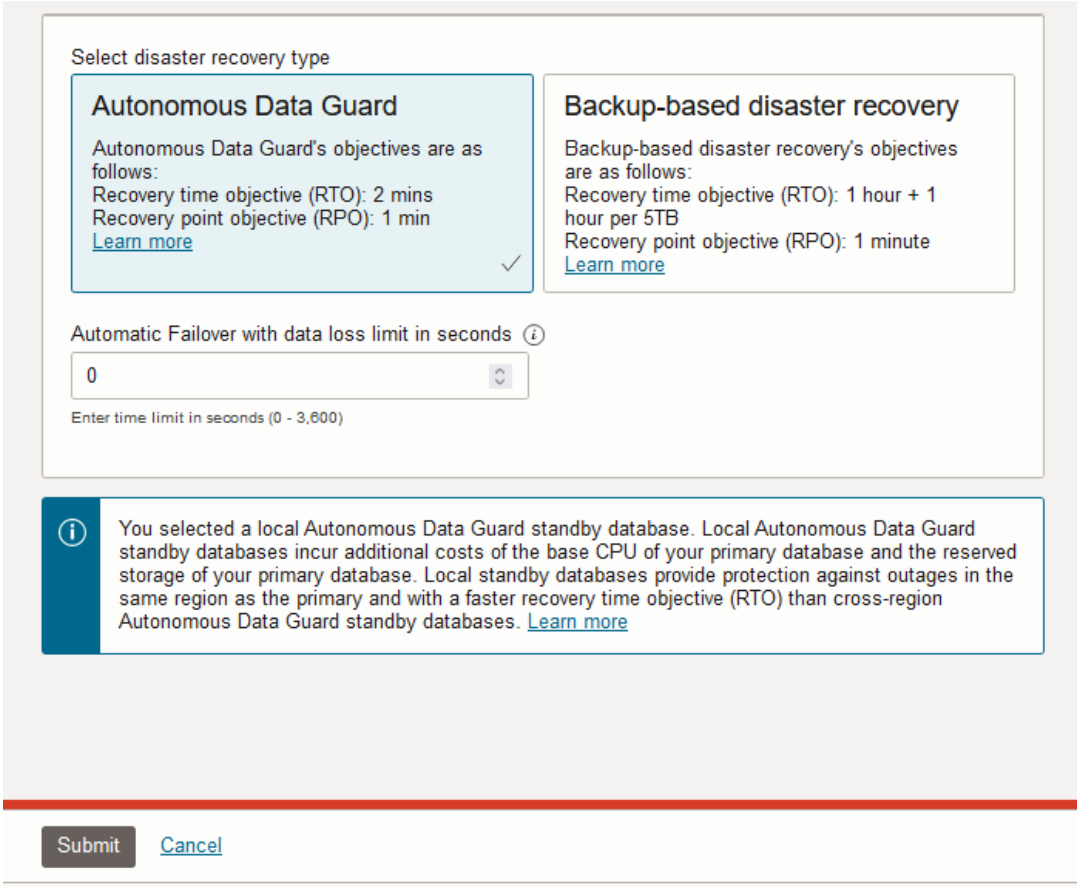
1. On the **Autonomous Database Details** page, in the **Resources** area select **Disaster recovery**.



2. In the row showing the database's disaster recovery details, click ⋮ at the end of the row and select **Update disaster recovery**.

   This shows the Update disaster recovery page.

3. Select Autonomous Data Guard.



4. In the **Automatic Failover with data loss limit in seconds** field, accept the default data loss limit of 0, or enter a custom value for the automatic failover data loss limit.

   See Automatic Failover with a Standby Database for more information.

5. Click **Submit**.

   The Autonomous Database Lifecycle State changes to **Updating**.

   The **State** column shows Provisioning while Autonomous Database provisions the standby database.

   After some time the **Lifecycle State** shows **Available** and the standby database provisioning continues.

   When provisioning completes the **DR Type** column shows Autonomous Data Guard.

> **Note:**
>
> While you add a new standby database, the primary database is available for read/write operations. There is no downtime on the primary database.

Notes for changing the disaster recovery type to a local Autonomous Data Guard standby database:

- Autonomous Database generates an Enable Autonomous Data Guard work request. To view the request, under **Resources** click **Work Requests**.

The work request may complete to 100% before you see Autonomous Data Guard in the **DR Type** column when you select **Disaster recovery** under Resources. The provisioning process takes several minutes.

- While you add a local standby database, when the **Lifecycle State** field shows **Updating**, the following actions are disabled for the primary database:
  - Move Resource. See Move an Autonomous Database to a Different Compartment for information on moving an instance.
  - Stop. See Stop Autonomous Database for information on stopping an instance.
  - Restart. See Restart Autonomous Database for information on restarting an instance.
  - Restore. See Restore and Recover your Autonomous Database for information on restoring.

Add a Cross-Region Standby Database

You can enable Autonomous Data Guard with a cross-region standby when Autonomous Database is available (Lifecycle State shows Available).

To add a standby database you must have adequate available resources. Adding an Autonomous Data Guard standby database will only be successful if adding the standby database does not cause you to exceed your Tenancy or compartment limits for CPU and Storage.

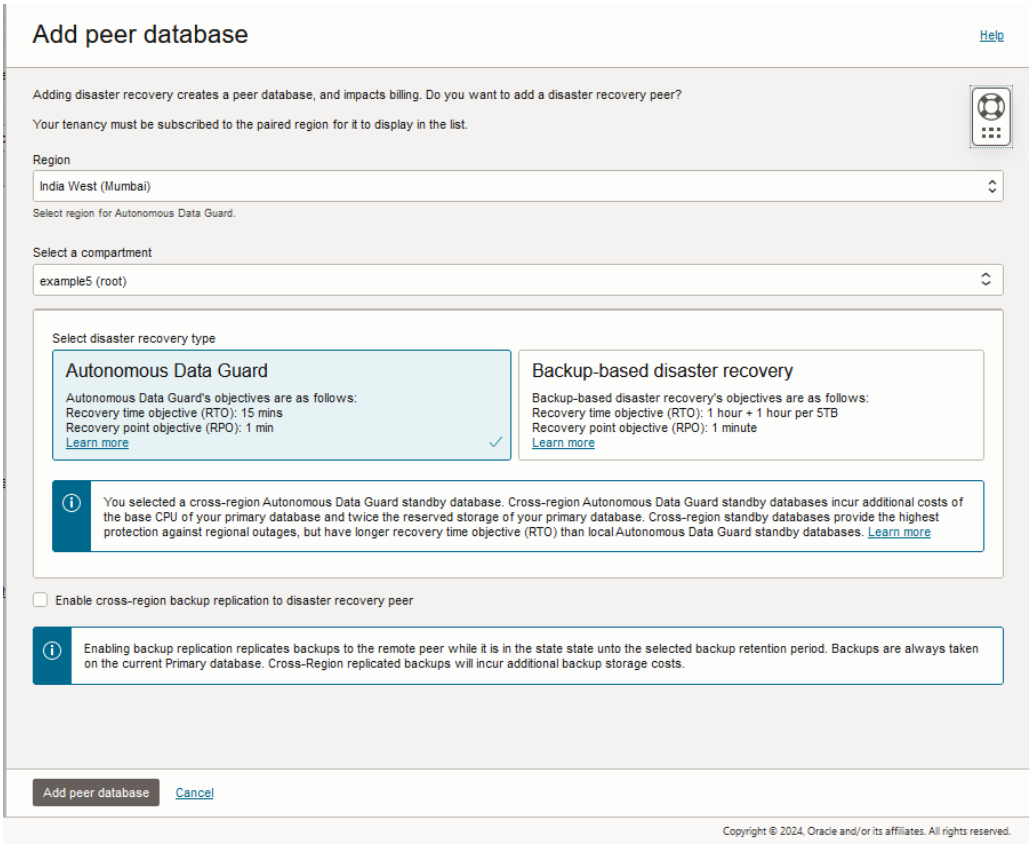
Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload, click one of: Autonomous Data Warehouse or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To add a cross-region Autonomous Data Guard standby:

1. On the **Autonomous Database Details** page, under **Resources**, click **Disaster recovery**.
2. Under Disaster recovery, click **Add peer database**.
3. In the **Region** field, select a region.

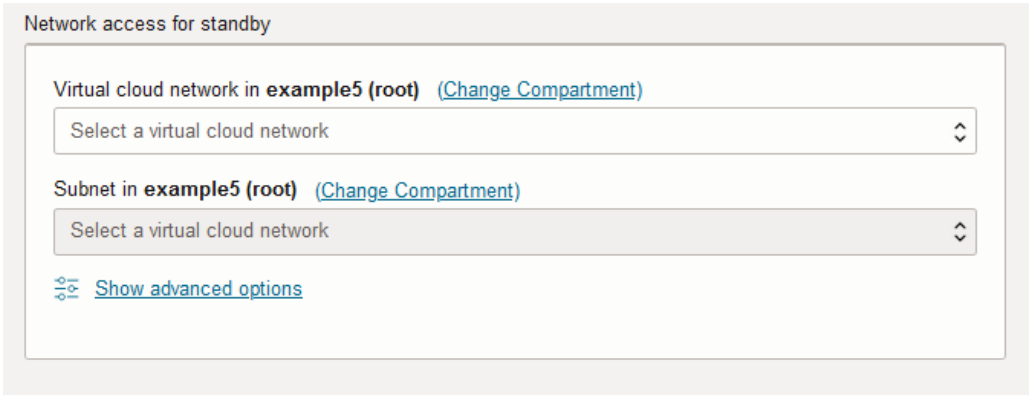
   The region list shows the available remote regions where you can create a cross-region standby. When you add a standby database, the list of available regions only shows a remote region if your tenancy is subscribed to the remote region (you must be subscribed to the paired remote region). See Autonomous Database Cross-Region Paired Regions for more information.
4. When you select a region, Autonomous Database shows the **Select a compartment** list. From this list, choose a compartment for the cross-region standby.
5. Select the disaster recovery type. In addition, when the source database is configured with a private endpoint, enter the private endpoint information for the peer.
   a. Select the disaster recovery type: **Autonomous Data Guard**.



b. If you want to enable cross-region backup replication, select **Enable cross-region backup replication to disaster recovery peer**

See Replicating Backups to a Cross-Region Autonomous Data Guard Standby for more information.

c. When the source database is configured with a private endpoint, in the **Network access for standby** area enter the **Virtual cloud network** and the **Subnet**.



In these Network access for standby fields you specify the private endpoint's VCN and Subnet on the remote region where the standby is created.

> **Note:**
>
> If you change your network access on the source database to enable a private endpoint after the standby is created, you must manually access the standby to enable a private endpoint on the peer.

6. Click **Add peer database**.

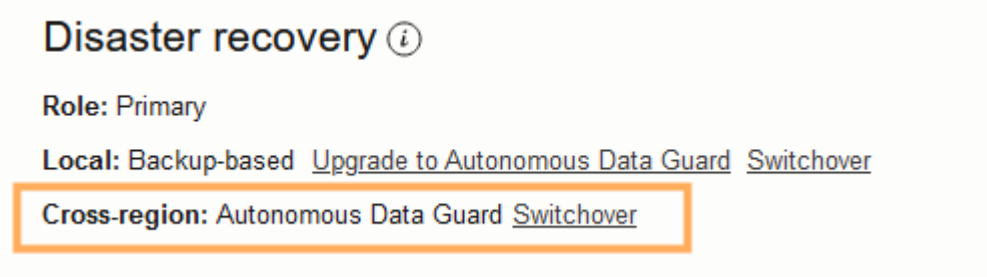
   The Autonomous Database **Lifecycle state** shows **Updating**. In the **Resources** area with **Disaster recovery** selected, the **State** field shows **Provisioning**.

   After some time, the **Lifecycle state** shows **Available** and the standby database provisioning continues.

> **Note:**
>
> While you add a standby database, the primary database is available for read/write operations. There is no downtime on the primary database.

When provisioning completes, on the **Autonomous Database Details** page under **Disaster recovery** the Oracle Cloud Infrastructure Console shows the following:

- **Role** shows **Primary**
- The **Local** field shows either **Backup-based** or **Autonomous Data Guard**. When the local field shows **Backup-based**, there is also a link, **Upgrade to Autonomous Data Guard**. Click this link to upgrade your local disaster recovery to Autonomous Data Guard.
- The **Cross-region** field shows **Autonomous Data Guard**.



When you enable a cross-region standby, the standby database created in the remote region has the same display name as the primary database with the extension: "_Remote".

When you click **Disaster recovery** under **Resources**, the **Peer Autonomous Database** column shows the standby database name and a provides a link. Click the link to go to the Oracle Cloud Infrastructure Console for the remote standby database.

Notes for adding a cross-region standby database:

- Autonomous Database generates the Enable cross-region disaster recovery work request. To view the request, under **Resources** click **Work requests**.
- After you add a cross-region (remote) standby database, the wallet and connection string from the primary database will contain only the hostname of the primary database, and the wallet and connection string from the remote database will contain only the hostname of the remote database. This applies for both instance and regional wallets.

  See Cross-Region Disaster Recovery Connection Strings and Wallets for more information.

- If you select **Enable cross-region backup replication to disaster recovery peer**, it can take between several minutes and several hours to replicate the backups to the remote region, depending on the size of the backups. After backups are replicated, when you select **Backups** under **Resources** on the peer database's Oracle Cloud Infrastructure Console, you will see the list of replicated backups.
- While you add a standby database and the **Lifecycle State** shows **Updating**, the following actions are disabled for the primary database:
  - Move Resource. See Move an Autonomous Database to a Different Compartment for information on moving an instance.
  - Stop. See Stop Autonomous Database for information on stopping an instance.
  - Restart. See Restart Autonomous Database for information on restarting an instance.
  - Restore. See Restore and Recover your Autonomous Database for information on restoring.
- See Cross-Region Autonomous Data Guard Notes and Notes for Customer-Managed Keys with Autonomous Data Guard for information on using customer-managed keys and for additional notes for using Autonomous Data Guard with a cross-region standby.
- Enable or Disable Backup Replication for an Existing Cross Region Standby

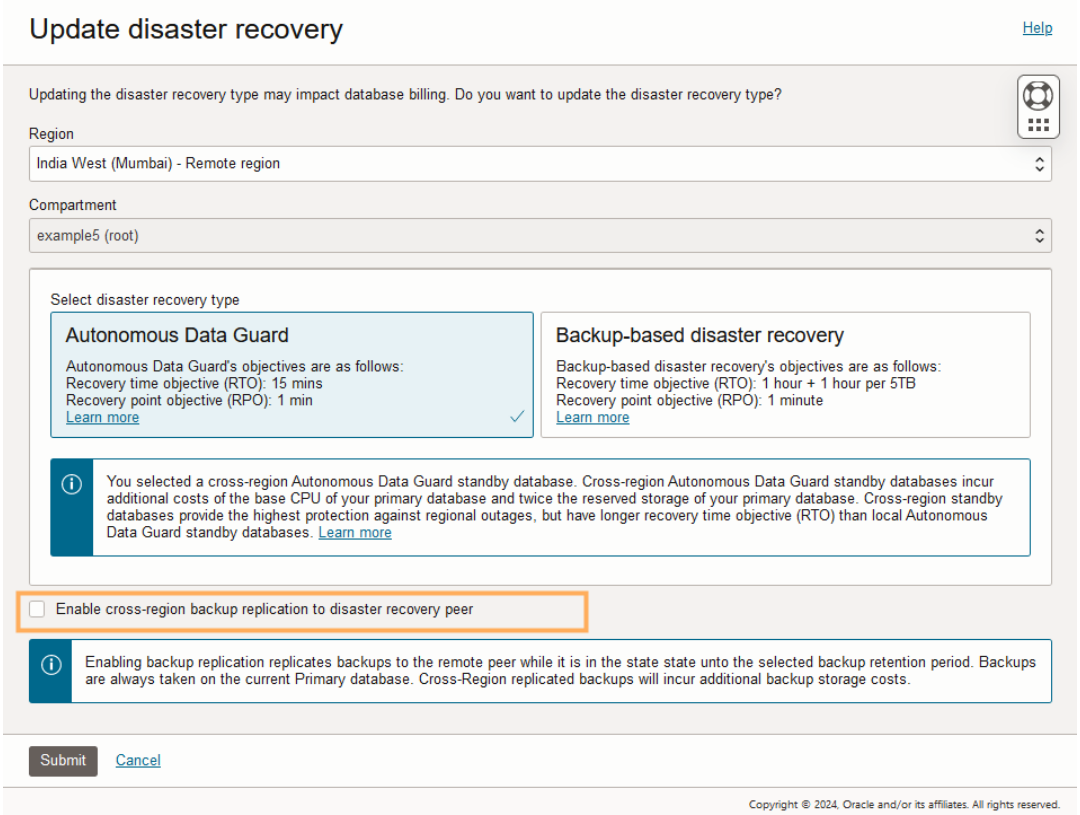
Enable or Disable Backup Replication for an Existing Cross Region Standby

You can enable or disable backup replication on a Autonomous Data Guard cross-region standby.

To enable or disable backup replication for an existing cross-region Autonomous Data Guard standby:

1. On the **Autonomous Database Details** page, in the **Resources** area select **Disaster recovery**.
2. In a row that lists a cross-region standby, click  at the end of the row and select **Update disaster recovery**.

   This shows the Update disaster recovery page.



3. Enable or disable backup replication.
   a. If cross-region backup replication is disabled, select **Enable cross-region backup replication to disaster recovery peer** to enable the option.
   b. If cross-region backup replication is enabled, deselect **Enable cross-region backup replication to disaster recovery peer** to disable the option.
4. Click **Submit**.

   The Autonomous Database Lifecycle State changes to **Updating**.

If you select **Enable cross-region backup replication to disaster recovery peer**, it can take between several minutes and several hours to replicate the backups to the remote region, depending on the size of the backups. After backups are replicated, when you select **Backups** under **Resources** on the peer database's Oracle Cloud Infrastructure Console, you will see the list of replicated backups.

Perform a Switchover

When you perform a switchover, the primary database becomes the standby database and the standby database becomes the primary database, with no data loss.

A switchover is typically done to test failover to the standby database for audit or certification reasons, or to test your application's failover procedures when you have added an Autonomous Data Guard standby database.

For switchover to a standby database, the Oracle Cloud Infrastructure Console on the primary database shows a **Switchover** link under **Disaster recovery** area when both the primary database and a standby database are available. You can perform a switchover when the primary database **Lifecycle State** shows **Available** or **Stopped**, and a standby database is available (the **State** field shows **Standby**).

To see either local or cross-region standby database state, under **Resources** click **Disaster recovery** and for the standby database listed in the **Peer Autonomous Database** column, check that the **State** shows **Standby**.

Using the Autonomous Database API, you can initiate the switchover operation at any time. See Use the API for more information.

- Perform a Switchover to a Local Standby
- Perform a Switchover to a Cross-Region Standby
- Notes for Performing a Switchover

Perform a Switchover to a Local Standby

When you perform a switchover, the primary database becomes the standby database and the standby database becomes the primary database, with no data loss.

To perform switchover to a local standby, you access the primary database from the Oracle Cloud Infrastructure Console.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload, click one of: Autonomous Data Warehouse or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To perform a switchover:

1. On the Autonomous Database Details page, under **Disaster recovery**, in the **Local** field, click **Switchover**.

   As an alternative, to initiate a switchover you can select **More actions** and **Switchover**.
2. In the **Confirm switchover to peer** dialog, enter the peer database name to confirm that you want to switch over.
3. In the **Confirm switchover to peer** dialog, click **Confirm switchover to peer**.

   When concurrent operations such as scaling are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

   The database **Lifecycle State** changes to **Updating**. To see the state of the peer database, under **Resources** click **Disaster recovery**. The state **State** shows **Role change in progress**.

When the switchover completes, Autonomous Database reports the time of the last switchover when you hover over the  in the **Role** field.

See Notes for Performing a Switchover for more information.

Perform a Switchover to a Cross-Region Standby

When you perform a switchover, the primary database becomes the standby database and the standby database becomes the primary database, with no data loss.

> **Note:**
>
> For a cross-region switchover you must initiate the switchover from the Standby database.

You have several options to access the Standby database:

- Select the remote region in Oracle Cloud Infrastructure Console and then access the Standby database.
- Access the primary, and from the primary database you can access the standby from the Autonomous Database Details page by selecting **Disaster recovery** under **Resources** and clicking the link for the standby database in the **Peer Autonomous Database** column.

To perform a switchover:

1. On the cross-region standby database, on the Autonomous Database Details page, under **Disaster recovery**, in the **Role** field, click **Switchover**.

   As an alternative, to initiate a switchover you can select **More actions** and **Switchover**.
2. In the **Confirm switchover to peer** dialog, enter the peer database name to confirm that you want to switch over.
3. In the **Confirm switchover to peer** dialog, click **Confirm switchover to peer**.

   When concurrent operations such as scaling are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

   The database **Lifecycle State** changes to **Updating**. To see the state of the peer database, under **Resources** click **Disaster recovery**. The state **State** shows **Role change in progress**.

When the switchover completes, Autonomous Data Guard does the following:

- The display name shows the standby indicator.
- The **Disaster recovery** resource information is updated to reflect the switchover. Under **Resources** select **Disaster recovery** to see the updated information.
- Autonomous Database reports the time of the last switchover when you hover over the  in the **Role** field.

See Notes for Performing a Switchover for more information.

Notes for Performing a Switchover

Notes for performing a switchover on Autonomous Database.

- For a cross-region switchover, you must initiate the switchover from the Standby database.
- During the switchover most of the actions on the Oracle Cloud Infrastructure Console are not available and the Autonomous Database Information page shows the **Lifecycle State** with the value **Updating**.

- The switchover operation keeps the original state of the primary database. If the primary database is stopped when you perform a switchover, the primary database is stopped after the switchover.
- Autonomous Database generates the Switchover Autonomous Database work request. To view the request, under **Resources** click **Work Requests**.
- After a switchover or failover to the standby, the standby becomes the primary:
  - The Oracle Cloud Infrastructure Metrics display information about the primary database.
  - The graphs on the **Database Dashboard** card in Database Actions display information about the primary database.
  - The graphs and metrics do not contain information about the database that was the primary database, before the switchover or failover operation.
- You cannot cancel a cross-region switchover operation after the switchover begins and the **State** shows **Role change in progress**. Your options are:
  - Try or retry a switchover or a failover until the operation succeeds.
  - File a service request at Oracle Cloud Support or contact your support representative.

Automatic Failover with a Standby Database

After you add a local Autonomous Data Guard standby database, the system monitors the primary instance and automatically fails over to a local standby database in certain scenarios.

If automatic failover is not possible, you have the option to perform a manual failover. Automatic failover does not apply to a cross-region standby.

Autonomous Database automatically fails over to a local standby database as follows:

- When the primary database becomes unavailable and users are not able to connect, Autonomous Data Guard automatically fails over to a local Autonomous Data Guard standby database if a local standby database is available, based on the Recovery Time Objective (RTO) and when Autonomous Data Guard determines that the automatic failover data loss limit will not be exceeded.
- Autonomous Data Guard performs automatic failover to a local standby database when a local standby database is available and the system can guarantee either the default zero data loss or up to the data loss limit you specified a data loss limit when you enabled Autonomous Data Guard. You can specify an automatic failover data loss limit between 0 and 3600 seconds. If this target cannot be met then automatic failover does not occur and you have the option to perform a manual failover.

  See Enable Autonomous Data Guard for information on setting the automatic failover data loss limit.
- After automatic failover to a local standby database completes, Autonomous Database creates a new local standby database for you.

  Autonomous Data Guard for a particular standby, either local or remote, is not enabled while the system is provisioning the new standby database. After Autonomous Data Guard completes the provisioning step for the standby database and it becomes available, you then have a new standby database with Autonomous Data Guard enabled on it.
- After automatic failover completes, Autonomous Database reports the time of the last failover when you hover over the ⓘ in the **Role** field.

- After an automatic Autonomous Data Guard failover, if there was a regional failure, when the region comes back online the standby database is automatically reconnected or if required reprovisioned.

If the primary database has failed or is unreachable and the conditions for Autonomous Data Guard automatic failover have not been met, then Oracle Cloud Infrastructure console shows a banner indicating that automatic failover did not succeed, provides a reason, such as possible data loss above the default 0 RPO or above the limit you set for the automatic failover data loss limit, and provides a link for you to initiate manual failover. See Perform Manual Failover to a Local Standby Database for more information.

> **Note:**
>
> Autonomous Data Guard automatic failover is disabled when the **Lifecycle State** is either: **Restore in Progress** or **Upgrading**.

See Autonomous Data Guard Recovery Time Objective (RTO) and Recovery Point Objective (RPO) for more information.

Perform a Manual Failover

When Autonomous Data Guard cannot automatically fail over to a local standby database, if a local standby database is available you can perform a manual failover to make the local standby database the primary database.

If a cross-region standby is available, you can perform a switchover to make the cross-region standby database the primary database. If the switchover fails, you can initiate a manual failover to the cross-region standby. It is possible for data loss to occur with a manual failover.

When you initiate a manual failover Autonomous Data Guard fails over to the standby database based on the Recovery Time Objective (RTO) and Recovery Point Objective (RPO) targets. See Autonomous Data Guard Recovery Time Objective (RTO) and Recovery Point Objective (RPO) for more information.

- Perform Manual Failover to a Local Standby Database
- Perform Manual Failover to a Cross-Region Standby Database
- Notes for Manual Failover with a Standby Database

Perform Manual Failover to a Local Standby Database

When Autonomous Data Guard cannot automatically fail over to a local standby database, if a local standby database is available you can perform a manual failover to make the local standby database the primary database.

When you add a local Autonomous Data Guard standby and automatic failover is not successful, Oracle Cloud Infrastructure console shows a banner with information about why the automatic failover was not successful. The Oracle Cloud Infrastructure console also shows a **failover** link in the **Role** field that you can click to initiate a manual failover to the local standby. The failover link only shows when the primary database is unavailable and a standby database is available. That is, the **Lifecycle State** field shows **Unavailable** and the local standby database is available.

Using the API, you can initiate manual failover at any time. See Use the API for information on using the API.

To see the standby database status, under **Resources** click **Disaster recovery** and for the standby database listed in the **Peer Autonomous Database** column, check that the **State** field shows **Available** or **Stopped**.

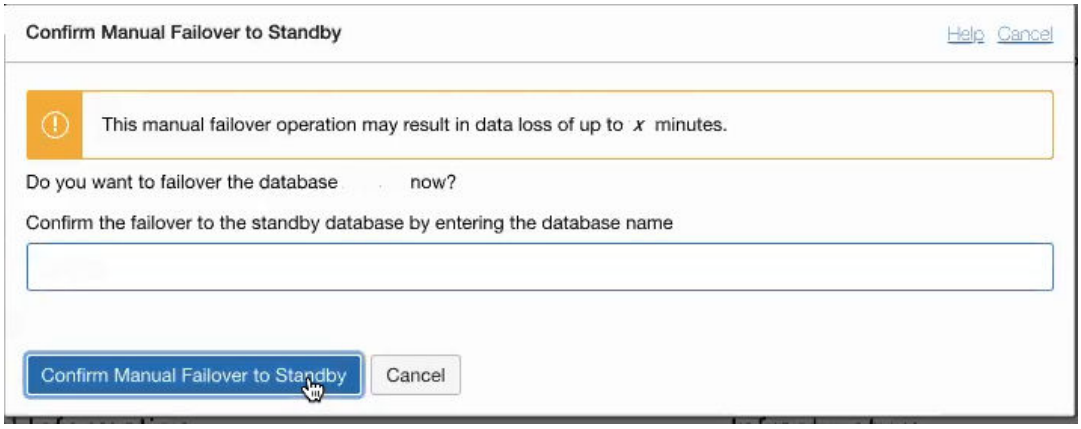
Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload, click one of: Autonomous Data Warehouse or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To initiate a manual failover when the primary database is unavailable and the local standby is available:

1. On the **Details** page, under **Disaster recovery**, in the **Local** field, click **Failover**.

   This shows the **Confirm Manual Failover to peer** dialog, along with information on possible data loss that may result if you perform the manual failover to standby.



2. In the **Confirm manual failover to peer** dialog, enter the Autonomous Database name to confirm that you want to failover.
3. In the **Confirm manual failover to peer** dialog, click **Confirm manual failover to peer**.

See Notes for Manual Failover with a Standby Database for information on steps that Autonomous Data Guard takes after failover completes.

Perform Manual Failover to a Cross-Region Standby Database

If a cross-region standby is available, you can perform a switchover to make the cross-region standby database the primary database. If the switchover fails, you can initiate a manual failover to the cross-region standby.

It is possible for data loss to occur with a manual failover. When you initiate a manual failover, Autonomous Database fails over to the standby database based on the Recovery Time Objective (RTO) and Recovery Point Objective (RPO) targets. See Autonomous Data Guard Recovery Time Objective (RTO) and Recovery Point Objective (RPO) for more information.

With both a local Autonomous Data Guard standby and a cross-region Autonomous Data Guard standby, when automatic failover is not successful and the local standby database is available, Oracle recommends that you attempt a manual failover to the local standby first (not to the remote standby).

If a local standby is unavailable or a manual failover to the local standby fails, you can perform a manual switchover to the cross-region standby. If the switchover to the cross-region standby fails, on the standby database the Oracle Cloud Infrastructure Console shows a **Failover** link in the **Role** field that you can click to initiate a manual failover to the standby database.

Using the API, you can initiate manual failover at any time. See Use the API for information on using the API.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload, click one of: Autonomous Data Warehouse or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To initiate a manual failover to a cross-region standby:

1. On the standby database, perform a switchover. See Perform a Switchover to a Cross-Region Standby for details.
2. If the switchover attempt in Step 1 fails, on the standby database the **Role** field shows a **Failover** link. On the standby database, click the **Failover** link.

   This shows the **Confirm manual failover to standby** dialog, along with information on possible data loss that may result if you perform the manual failover to the standby database.
3. In the **Confirm manual failover to standby** dialog, enter the Autonomous Database name to confirm that you want to failover.
4. In the **Confirm manual failover to standby** dialog, click **Confirm manual failover to standby**.

   When concurrent operations such as scaling are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

See Notes for Manual Failover with a Standby Database for information on steps that Autonomous Data Guard takes after failover completes.

Notes for Manual Failover with a Standby Database

When the failover completes, Autonomous Data Guard performs post failover steps.

- For a failover to the local standby, Autonomous Data Guard creates a new local Standby database after the failover. Autonomous Data Guard is not enabled when the system is provisioning the new Standby database and the **Role** field shows **Provisioning**. After Autonomous Data Guard completes the provisioning step, then you have a new local Standby database and Autonomous Data Guard is enabled.
- For a failover to a cross-region standby, if there was a local standby before the failover, Autonomous Data Guard makes the local standby available again for a switchover or failover on the current Primary.
- After a manual failover operation completes, you can see any data loss associated with the manual failover in the message on the Oracle Cloud Infrastructure console banner. The manual failover data loss is specified in minutes.

  This information also shows when you hover over the  in the **Role** field.

- After a manual Autonomous Data Guard failover, if there was a regional failure, when the region comes back online the standby database is automatically reconnected or if required reprovisioned.

Convert Cross Region Peer to Snapshot Standby

A cross-region peer can be converted to a snapshot standby. This converts the peer to a read-write database for up to two days.

Snapshot standby CPU usage is billed based on the base CPU count and any additional CPU usage if compute auto scaling is enabled. The number of base CPUs is specified by the number of ECPUs (OCPUs if your database uses OCPUs) as shown in the **ECPU count** or **OCPU count** field on the Oracle Cloud Infrastructure Console.

Snapshot standby storage usage is billed based on the storage of the snapshot standby plus 1 x the storage of the source primary database.

- About Disaster Recovery Snapshot Standby Databases
- Convert Cross Region Disaster Recovery Peer to a Snapshot Standby
  You can convert a cross-region disaster recovery peer to a snapshot standby.
- Convert Snapshot Standby Back to a Cross-Region Disaster Recovery Peer
  You can manually convert a snapshot standby back to a disaster recovery peer for the primary (source database). After the converting back, the snapshot standby returns to its role as a disaster recovery standby.

About Disaster Recovery Snapshot Standby Databases

Converting a disaster recovery peer to a snapshot standby opens the database in read-write mode and the cross-region disaster recovery peer temporarily stops refreshing data from the source database.

While operating as a snapshot standby, updates from the source database are still sent to the snapshot standby, and you are protected if the source database region were to encounter a failure, however the updates are not applied to the snapshot standby until the database is converted back to a disaster recovery peer.

- Snapshot Standby Features
  Provides information on snapshot standby features.
- Snapshot Standby Operations
  After you create a snapshot standby you can perform almost all database operations on the snapshot standby. There are some operations that are not allowed on a snapshot standby.
- Snapshot Standby Reconnect Time
  A banner on the Oracle Cloud Infrastructure Console indicates the date and time when the snapshot standby automatically reconnects to the source database. At the time indicated on the banner, Autonomous Database converts a snapshot standby back to the **Standby** role.

Snapshot Standby Features

Provides information on snapshot standby features.

While the database is in the **Snapshot standby** role:

- By converting a cross-region disaster recovery peer, you can use the snapshot standby for testing and querying the data in the peer. This allows you to test with no downtime on the primary (source) database, compared to testing by using switchover to the remote peer.
- You can use a snapshot standby to completely test your disaster recovery environment, including making any changes required to verify your standby environment, such as the mid-tier configuration. Using a snapshot standby you can make configuration changes or perform DML operations on the database as needed for complete testing and verification of the standby environment.

Reconnecting a snapshot standby to the primary (source database):

- Reconnect to the primary, source database, when you are done with the tasks that require the snapshot standby to be open for read-write operations. If you do not manually reconnect within two days, the snapshot standby automatically reconnects to the primary.
- Oracle recommends you convert a snapshot standby back to a disaster recovery peer as soon as you are done with the operations that require the standby to be open for read-write operations. When you convert back to a to a disaster recovery peer, the accumulated changes from the source database are applied on the peer. If you keep the disaster recovery peer open as a snapshot standby for a longer period, assuming there are ongoing changes on the primary during this time, it takes longer to convert back to a disaster recovery peer.

When the snapshot standby reconnects to the primary database, Autonomous Database performs the following actions:

- The disaster recovery type you were using, and any associated billing returns to the type it was before you performed the convert disaster recovery peer to snapshot standby. That, the disaster recovery peer returns to the same type of disaster recovery peer, either Backup-Based Disaster Recovery (**Backup copy**) or **Autonomous Data Guard**, as shown in the DR Type column in the **Disaster Recovery** area.
- Any changes on the snapshot standby from the time it was converted to a snapshot standby, until the time it reconnects to the source are discarded. This means all changes, including metadata, that is inserted, updated, or deleted while the database operates as a snapshot standby are lost (discarded) when the snapshot standby reconnects to its source database.
- All changes that occurred on the primary are replicated to the remote region while the database operates as a snapshot standby, but the changes are not applied to the snapshot standby. The changes that occur on the primary during this period are applied to the snapshot standby when it is converted back to a disaster recovery peer.
- On the Oracle Cloud Infrastructure Console, the role updates from **Role: Snapshot standby** to **Role: Standby** or **Role: Backup copy**, depending on the disaster recovery type.

Snapshot Standby Operations

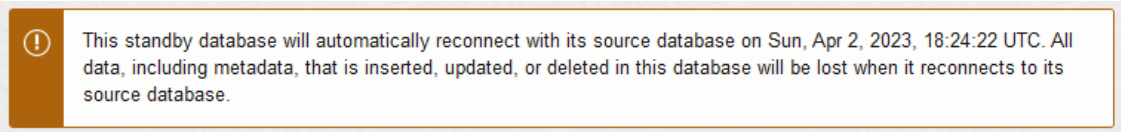
After you create a snapshot standby you can perform almost all database operations on the snapshot standby. There are some operations that are not allowed on a snapshot standby.

Operation	Description
Convert to Snapshot Standby	You can convert a cross-region peer to a snapshot standby. See Convert Cross Region Disaster Recovery Peer to a Snapshot Standby for the steps to convert a peer database to snapshot standby.

Operation	Description
Start or Restart	When a snapshot standby is stopped, as indicated by the Lifecycle State **Stopped**, you can start the database. When a snapshot standby is available, as indicated by the Lifecycle State **Available**, you can restart the database or stop the database.
Convert Snapshot Standby Back to Disaster Recovery Peer	When a snapshot standby is in the **Snapshot standby** role, the database operates as a read-write database. A snapshot standby has a two day (48 hour) limit up to which it can stay in the **Snapshot standby** role. If you do not manually convert the snapshot standby back within two days, the snapshot standby automatically converts back to a disaster recovery peer. See Convert Snapshot Standby Back to a Cross-Region Disaster Recovery Peer for more information.
Stop	When a snapshot standby is stopped, database operations are not available and charging for CPU usage on the snapshot standby stops.
Terminate	You are not allowed to terminate a snapshot standby. You can reconnect the snapshot standby to the primary. See Convert Snapshot Standby Back to a Cross-Region Disaster Recovery Peer for more information.
Create Refreshable Clone	You are not allowed to create a refreshable clone on a snapshot standby.
Disaster Recovery Peer	You are not allowed to add an Autonomous Data Guard standby database or a Backup-Based Disaster Recovery peer to a snapshot standby.

Snapshot Standby Reconnect Time

A banner on the Oracle Cloud Infrastructure Console indicates the date and time when the snapshot standby automatically reconnects to the source database. At the time indicated on the banner, Autonomous Database converts a snapshot standby back to the **Standby** role.



> **Note:**
>
> When a snapshot standby is not reconnected within 48 hours, the snapshot standby automatically reconnects to the source database.

Convert Cross Region Disaster Recovery Peer to a Snapshot Standby

You can convert a cross-region disaster recovery peer to a snapshot standby.

> **Note:**
>
> All data, including metadata, that is inserted, updated, or deleted in the database during the disconnect period will be lost when a snapshot standby reconnects to its source database. All changes that occur on the primary during the disconnect period are applied to the standby when it reconnects to the source database.

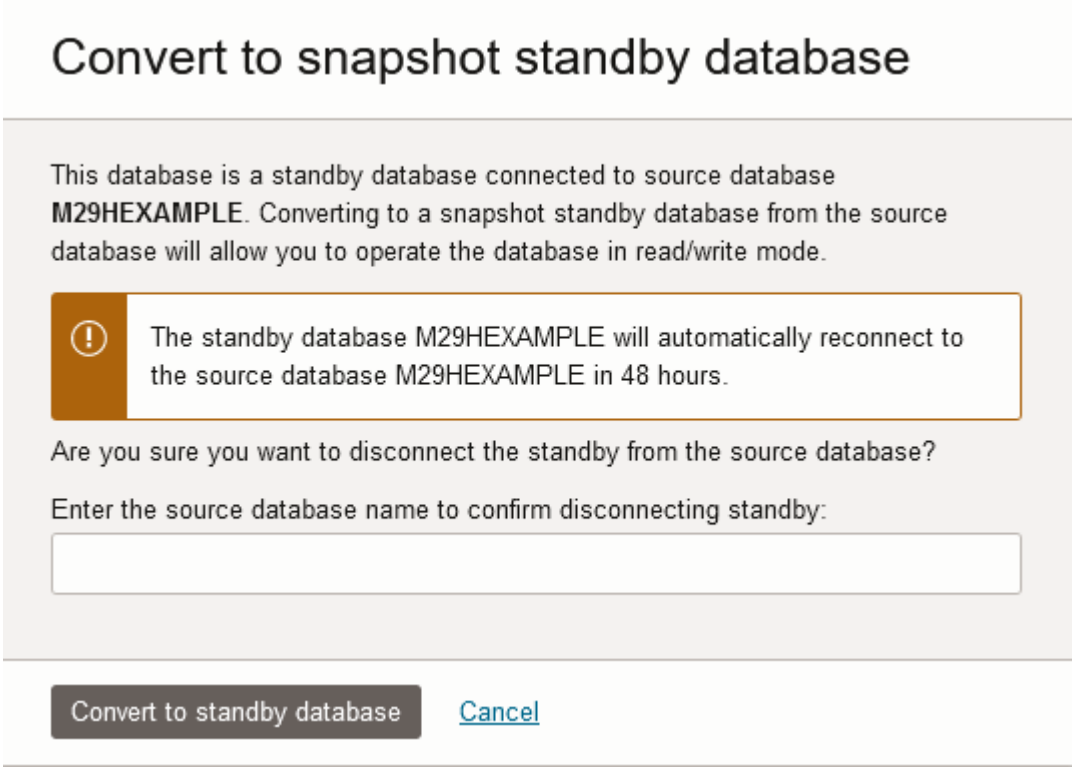
Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload, click one of: Autonomous Data Warehouse or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

1. On the Primary Region Autonomous Database instance, on the Autonomous Database Details page under **Resources**, select **Disaster recovery**.
2. Access the remote region peer.

   On the Primary Region Autonomous Database instance the **Disaster recovery** information area shows the **Peer Autonomous Database** column.

   Under **Peer Autonomous Database** column, click the link to access the cross-region peer.
3. On the cross-region peer, from the **More actions** drop-down list, select **Convert to snapshot standby database**.



4. On the **Convert to snapshot standby database** page, enter the source database name to confirm the disconnect.
5. Click **Convert to snapshot standby database**.

   The Autonomous Database Lifecycle State changes to **Updating**.

> **Note:**
>
> While you convert to a snapshot standby, the primary database is available for read/write operations. There is no downtime on the primary database.

When the operation completes, note the following:

- On the snapshot standby, there is a banner that indicates the date and time when the standby database will automatically reconnect with its source database
- On the snapshot standby, on the Autonomous Database details page, under **Disaster recovery**, the **Role** shows **Snapshot standby**.
- On the snapshot standby's source database, on the Autonomous Database details page, when you click **Disaster recovery** under **Resources**, the **Peer role** shows **Snapshot standby**.
- On the snapshot standby, you can scale CPU or storage, independent of the source database.
- When you scale CPU or storage on the primary, the changes to the primary do not affect the snapshot standby until it is converted back to a disaster recovery peer.

Convert Snapshot Standby Back to a Cross-Region Disaster Recovery Peer

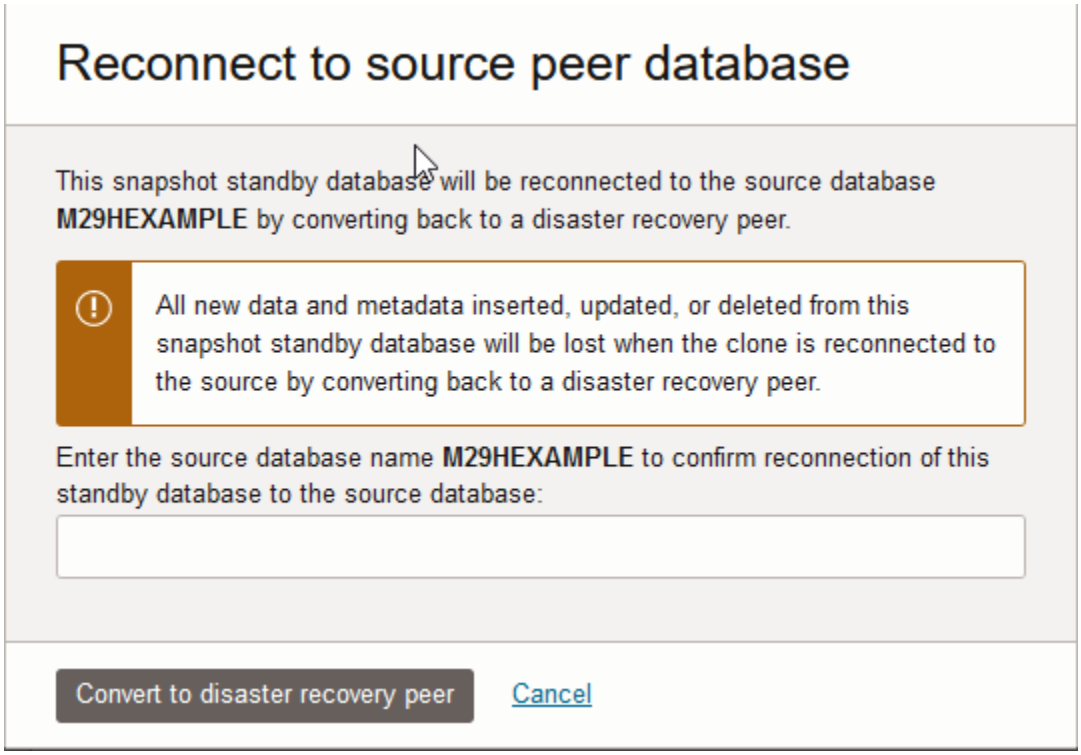
You can manually convert a snapshot standby back to a disaster recovery peer for the primary (source database). After the converting back, the snapshot standby returns to its role as a disaster recovery standby.

> **Note:**
>
> All data, including metadata, that is inserted, updated, or deleted in the snapshot standby database during the disconnect period will be lost when the snapshot standby reconnects to its source database.
> All changes on the primary that were sent to the snapshot standby but not applied during the disconnect period will be applied to the standby when it reconnects to the source database.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload, click one of: Autonomous Data Warehouse or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

1. On the snapshot standby, from the **More actions** drop-down list, select **Reconnect to source peer database**.



2. On the **Reconnect to source peer database** page, enter the source database name to confirm the reconnect.
3. Click **Convert to disaster recovery peer**.

   The Autonomous Database Lifecycle State changes to **Updating**.

> **Note:**
>
> While you reconnect the standby to the source database, the primary (source) database is available for read/write operations. There is no downtime on the primary database.

When the snapshot standby reconnects to the primary database, Autonomous Database does the following:

- The disaster recovery type you were using, and any associated billing returns to the type it was before you performed the convert disaster recovery peer to snapshot standby. That, the disaster recovery peer returns to the same type of disaster recovery peer, either Backup-Based Disaster Recovery (**Backup copy**) or **Autonomous Data Guard**, as shown in the DR Type column in the **Disaster Recovery** area.
- Any changes on the snapshot standby from the time it was converted to a snapshot standby, until the time it reconnects to the source are discarded. This means all changes, including metadata, that is inserted, updated, or deleted while the database operates as a snapshot standby are lost (discarded) when the snapshot standby reconnects to its source database.
- All changes that occurred on the primary are replicated to the remote region while the database operates as a snapshot standby, but the changes are not applied to the snapshot standby. The changes that occur on the primary during this period are applied to the snapshot standby when it is converted back to a disaster recovery peer.

- On the Oracle Cloud Infrastructure Console, the role updates from **Role: Snapshot standby** to **Role: Standby** or **Role: Backup copy**, depending on the disaster recovery type.

Update Standby to Use a Backup Copy Peer

Describes the steps to change the disaster recovery type from Autonomous Data Guard standby database to Backup-Based Disaster Recovery.

These steps terminate the standby database. By default Autonomous Database provides a Backup-Based Disaster Recovery peer and these steps change the disaster recovery type from Autonomous Data Guard to backup copy.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload, click one of: Autonomous Data Warehouse or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To change the disaster recovery type:

1. On the Autonomous Database Details page, under **Resources**, select **Disaster recovery**.
2. In the row showing the disaster recovery details, click the Actions icon (three dots) and select **Update DR type**.

   This shows the Update disaster recovery type page.
3. Select **Backup-based disaster recovery**.
4. Click **Submit**.
5. In the Disable Autonomous Data Guard dialog, click **Disable Autonomous Data Guard**.

   While the standby database is terminating, the **Lifecycle State** changes to **Updating**.

   Autonomous Database generates the Disable Autonomous Data Guard work request. To view the request, under **Resources**, click **Work Requests**.

Disable a Cross Region Standby Database

Disabling a cross-region Autonomous Data Guard standby database terminates the standby database. If you later add a cross-region Autonomous Data Guard standby, the system creates a new cross-region standby database.

You have two options to disable a cross-region Autonomous Data Guard standby:

You can update the cross-region disaster recovery type to use Backup-Based Disaster Recovery. This terminates the cross-region Autonomous Data Guard standby database and adds a cross-region backup copy peer.

You can terminate the cross-region Standby database.

- Update Cross-Region Standby to Use Backup-Based Disaster Recovery
  You can update the disaster recovery type from cross-region Autonomous Data Guard standby database to cross-region Backup-Based Disaster Recovery. This terminates the cross-region Autonomous Data Guard standby database.

- Terminate a Cross-Region Standby Database
  Describes the steps to terminate a cross-region standby database.
- Verify Cross-Region Autonomous Data Guard is Disabled
  Describes the steps to verify that an Autonomous Data Guard cross-region standby is disabled.

Update Cross-Region Standby to Use Backup-Based Disaster Recovery

You can update the disaster recovery type from cross-region Autonomous Data Guard standby database to cross-region Backup-Based Disaster Recovery. This terminates the cross-region Autonomous Data Guard standby database.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload, click one of: Autonomous Data Warehouse or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To change your disaster recovery type to use a cross-region backup-copy peer:

1. On the Primary Role Autonomous Database, on the Autonomous Database Details page under **Resources**, select **Disaster recovery**.
2. Access the remote standby.

   On the primary Autonomous Database instance the **Disaster recovery** information area shows the remote peer. The remote peer has the same display name as the primary database, with an "`_Remote`" extension.

   Under **Peer Autonomous Database**, click the link to access the cross-region peer.
3. On the remote standby database, click **Update disaster recovery** under **Disaster recovery** on the Oracle Cloud Infrastructure Console.
4. On the Update disaster recovery type page, select **Backup-based disaster recovery**.
5. Click **Submit**.

   The Autonomous Database Lifecycle State changes to **Updating**.

> **Note:**
>
> While you update the disaster recovery type, the primary database is available for read/write operations. There is no downtime on the primary database.

Autonomous Database generates a Change Disaster Recovery Configuration work request. To view the request, under **Resources** click **Work Requests**.

Terminate a Cross-Region Standby Database

Describes the steps to terminate a cross-region standby database.

Perform the following prerequisite steps as necessary:

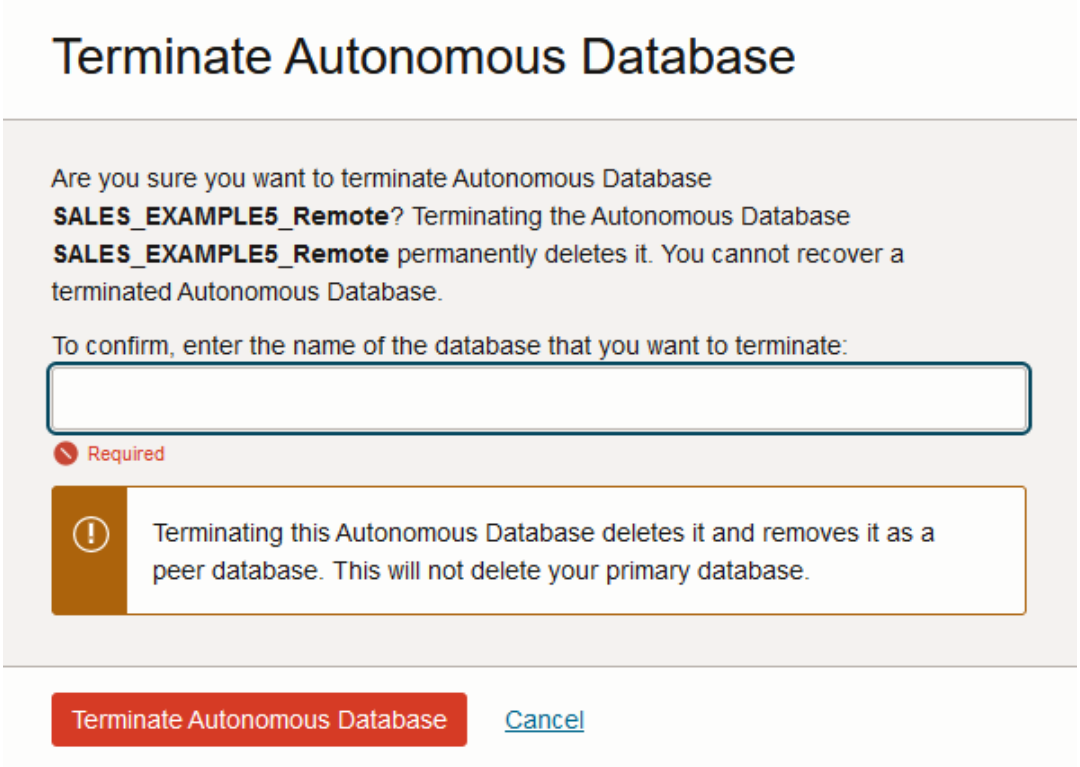
- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload, click one of: Autonomous Data Warehouse or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To terminate a cross-region standby database:

1. On the primary database, on the Autonomous Database Details page, under **Resources** select **Disaster recovery**.
2. Access the remote standby.

   The **Disaster recovery** information area shows the **Peer Autonomous Database**. The remote standby database has the same display name as the primary database, with an "`_Remote`" extension.

   Under **Peer Autonomous Database**, click the link to access the cross-region peer.
3. On the console for the remote standby database, from the **More actions** drop-down list, select **Terminate**.



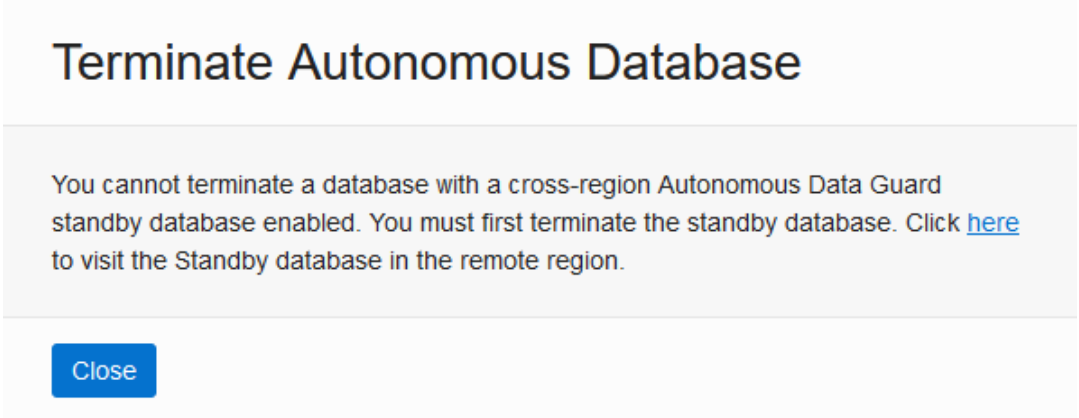
4. On the **Terminate Autonomous Database** page enter the database name to confirm that you want to terminate the cross-region standby database.
5. Click **Terminate Autonomous Database**.

   While the standby database is terminating, the **Lifecycle State** changes to **Terminating**.

> **Note:**
>
> If there is a local Autonomous Data Guard standby database, terminating the cross-region standby does not change the local disaster recovery option.

Note the following when your disaster recovery option includes a cross-region standby:

- A cross-region standby database cannot be terminated from the primary database.
- When there is a cross-region standby database, you must terminate the cross-region standby before you terminate the primary database. If you attempt to terminate the primary, the system shows the following message:



  In this case, after you terminate the cross-region (remote) standby you can terminate the Primary role database.

After Autonomous Data Guard is disabled, terminate the database. See Terminate an Autonomous Database Instance for more information.

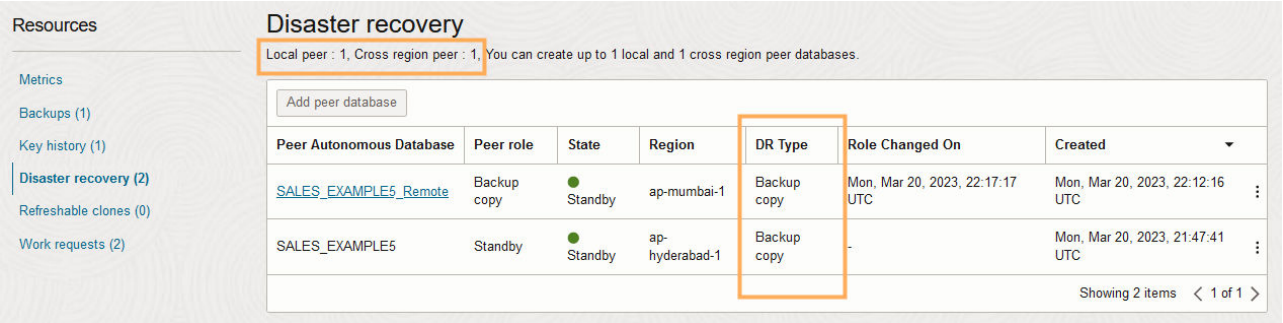
Verify Cross-Region Autonomous Data Guard is Disabled

Describes the steps to verify that an Autonomous Data Guard cross-region standby is disabled.

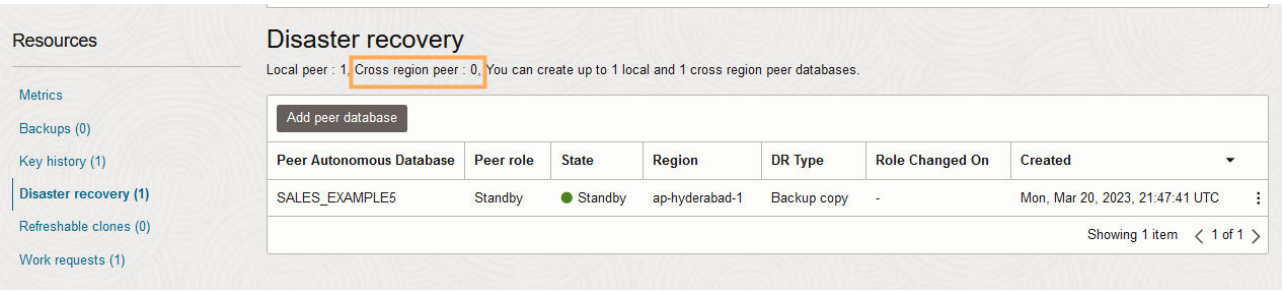
To verify that a cross-region Autonomous Data Guard standby database is disabled:

1. On the primary database, on the Autonomous Database Details page under **Resources**, select **Disaster recovery**.
2. Verify that either the disaster recovery type is changed to backup copy or the cross-region Autonomous Data Guard standby is terminated.

- If you updated the disaster recovery type to use cross-region backup based disaster recovery, verify that the remote peer autonomous database **DR Type** shows **backup copy**.



- If you manually terminated standby database, verify that the cross-region standby count is 0 (that is there is no remote peer).



Update Remote Peer Network ACLs

You can independently modify network ACLs on a remote disaster recovery peer database.

By default the disaster recovery primary and remote peer databases use the same network Access Control Lists (ACLs). Optionally, you can configure ACLs independently on remote peer databases. This provides an option to use different ACLs on remote peer databases.

If you modify the ACLs on a remote peer, Autonomous Database no longer keeps the ACL configuration synchronized between the primary and the remote peer. After you modify the ACLs on a remote peer, Autonomous Database manages the ACLs on the remote peer independently.

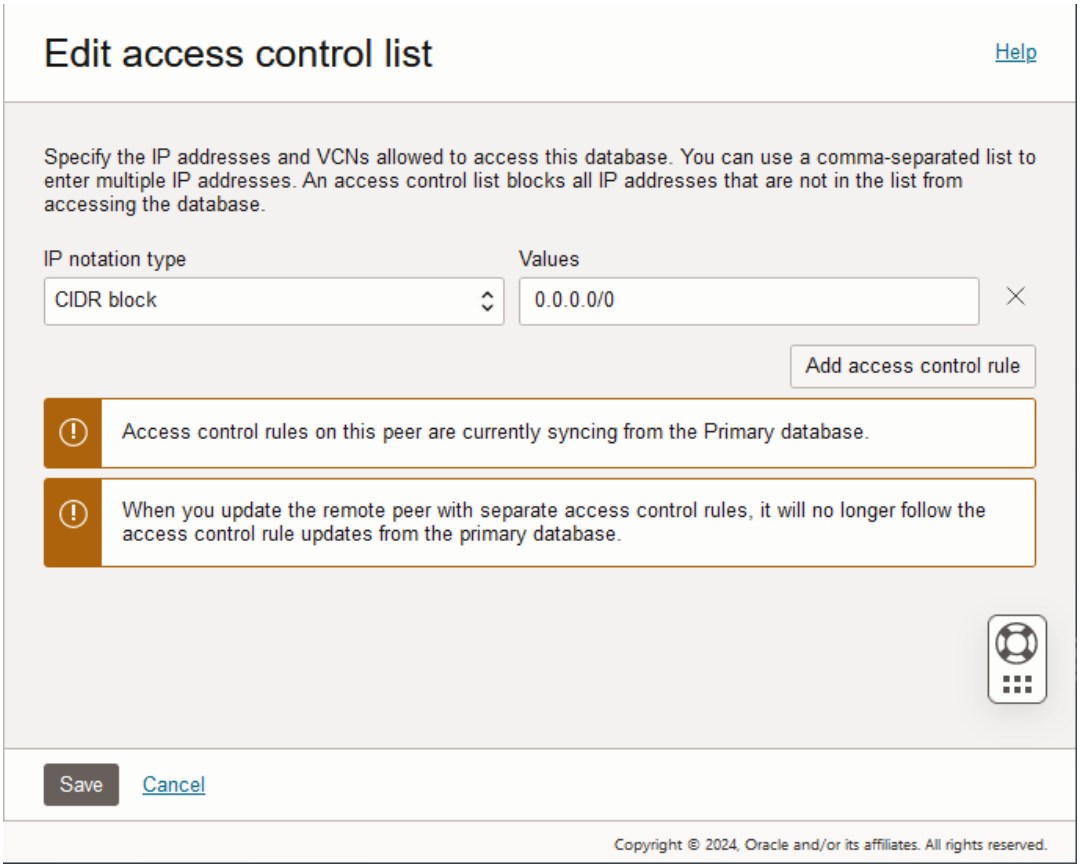
To use different network ACLs on a remote Autonomous Database peer:

1. On the primary database, on the Autonomous Database Details page, under **Resources** select **Disaster recovery**.
2. Access the remote peer.

   The **Disaster recovery** information area shows the **Peer Autonomous Database**. The remote peer database by default has the same display name as the primary database, with an extension. For example, `DBNAME_remote`.

   Under **Peer Autonomous Database**, click the link to access a cross-region peer.
3. On the remote peer database, edit the access control list.

   Before you change ACL values the dialog shows a message indicating that ACLs on the peer database are syncing from the Primary database. For example:



See Configure Access Control Lists for an Existing Autonomous Database Instance for more information.

4. Add, remove, or modify one or more ACLs.
5. Click **Save**.

After you modify ACLs, the ACLs on the primary and on the remote peer are managed separately.

If you want to restart the synchronization of ACLs between the primary and the remote peer, you have two options:

- Terminate the peer Autonomous Database and create a new cross region disaster recovery peer database.

  See Disable a Cross Region Standby Database for details on terminating a remote standby database.

  See Disable a Cross-Region (Remote) Peer for details on terminating a remote peer.
- Contact Oracle Cloud Support and file a service request or contact your support representative.

Events and Notifications for a Standby Database

You can use Oracle Cloud Infrastructure Events to be notified of and to specify rules to automatically respond to Autonomous Data Guard operations.

> **Note:**
>
> When Autonomous Data Guard is enabled with a remote standby, events are only sent to the instance with the **Primary** role.

Event Description	Type
Automatic Failover Begin	`com.oraclecloud.databaseservice.automaticfailover.begin`
Automatic Failover End	`com.oraclecloud.databaseservice.automaticfailover.end`
Disable Data Guard Begin	`com.oraclecloud.databaseservice.disableautonomousdataguard.begin`
Disable Data Guard End	`com.oraclecloud.databaseservice.disableautonomousdataguard.end`
Enable Data Guard Begin	`com.oraclecloud.databaseservice.enableautonomousdataguard.begin`
Enable Data Guard End	`com.oraclecloud.databaseservice.enableautonomousdataguard.end`
Manual Failover Begin	`com.oraclecloud.databaseservice.failoverautonomousdatabase.begin`
Manual Failover End	`com.oraclecloud.databaseservice.failoverautonomousdatabase.end`
Switchover Begin	`com.oraclecloud.databaseservice.switchoverautonomousdatabase.begin`
Switchover End	`com.oraclecloud.databaseservice.switchoverautonomousdatabase.end`

See the following Oracle Cloud Infrastructure topics for information on events and notifications:

- Overview of Events
- Autonomous Database Event Types
- Notifications Overview

Use the API

Provides links for details on using API operations to manage Autonomous Data Guard.

For information about using the API and signing requests, see REST APIs and Security Credentials.

For information about SDKs, see Software Development Kits and Command Line Interface.

Use these API operations to manage Autonomous Data Guard:

- To enable or disable Autonomous Data Guard, use UpdateAutonomousDatabase.
- To initiate a manual failover operation, use FailOverAutonomousDatabase.
- To initiate a switchover operation, use SwitchOverAutonomousDatabase.

Use these Terraform APIs to manage Autonomous Database resources:

For information about Terraform, see Terraform Provider and for information about Terraform APIs, see Data Source: oci_database_autonomous_database.

Autonomous Data Guard Notes

Note the following for using Autonomous Database with an Autonomous Data Guard standby database:

- You cannot connect to a Standby database until it is made the Primary by a failover or a switchover. Thus, a Standby database cannot be opened for read-only access and cannot be used to offload queries from a Primary database.
- Autonomous Data Guard is available with the Data Warehouse and Transaction Processing workload types. Autonomous Data Guard is not available with the JSON and APEX workload types.
- Autonomous Data Guard is not available with Always Free Autonomous Databases.
- Autonomous Database does not provide access to a local Standby database:
  - You perform all operations, such as scaling up the **ECPU count** (**OCPU count** if your database uses OCPUs) enabling compute auto scaling on the Primary database and Autonomous Database performs the same actions on the local Standby database. Likewise, you only perform actions such as stopping or restarting the database on the Primary database.
  - A local Standby database is not available for use as a read-only database.
- Autonomous Database provides access to a remote Standby database:
  - You perform most operations, such as scaling up the **ECPU count** (**OCPU count** if your database uses OCPUs) and enabling compute auto scaling on the Primary database and Autonomous Database performs the same actions on the remote Standby database. Likewise, you only perform actions such as stopping or restarting the database on the Primary database.
  - You can perform certain operations, such as configuring private endpoints on a remote Standby database.
  - A remote Standby database is not available for use as a read-only database.
- The Number of ECPUs (OCPUs if your database uses OCPUs) allocated graph and the CPU utilization graph on the **Database Dashboard** card in Database Actions displays the ECPUs (OCPUs if your database uses OCPUs) allocated and the CPUs utilization for the

Primary database. These graphs do not include information about a local Standby database or about a remote Standby database.

The CPU Utilization metrics on the Oracle Cloud Infrastructure Console metrics page display the CPU utilization for the Primary database. Other metrics on this page also apply to the Primary database. These metrics do not include information about the local Standby database or remote Standby database.

- After a switchover or failover to the Standby, the Standby becomes the Primary and the graphs on the **Database Dashboard** in Database Actions and the Oracle Cloud Infrastructure Console metrics page display information about the Primary database. The graphs and metric do not contain information about the database that was the Primary before the switchover or failover operation.
- Automatic Failover to a local Standby is disabled during a Restore in Progress operation.
- Automatic Failover to a local Standby disabled when Upgrading a Database.
- When the **Lifecycle State** field for the Primary database shows **Stopped**, the Standby database is also stopped. You may still perform a switchover when the Primary database is **Stopped**.
- Cross-Region Autonomous Data Guard Notes
  Note the following for using Autonomous Database with an Autonomous Data Guard remote standby database:

Cross-Region Autonomous Data Guard Notes

Note the following for using Autonomous Database with an Autonomous Data Guard remote standby database:

The following are restrictions and limitations when you enable Autonomous Data Guard with a remote Standby database:

- To disable Autonomous Data Guard with a cross-region standby database, you terminate the remote Standby database. See Terminate a Cross-Region Standby Database for more information.
- When a private endpoint is enabled or disabled on the Primary database, any previously configured Access Control List (ACL) on the Standby is enabled and the values are cleared. You must reset and verify the ACL on the Standby database after you disable a private endpoint on the Primary.
- Oracle Data Safe can be enabled on a database that has a cross-region Standby database enabled, but it only monitors the database within its region, and cannot monitor the standby in the event of a switchover or a failover.
- When you allow TLS authentication for the Primary database, Autonomous Data Guard enables TLS authentication in the cross- region Standby. Thus, when Autonomous Data Guard is enabled with a remote Standby, you can only allow TLS connections on the Primary if both the Primary and the remote Standby are configured to support TLS connections. That is, the Primary and the remote Standby must either be configured with ACLs or with a private endpoint. See Network Configuration Prerequisites to Allow TLS Authentication for more information.
- See for the following information on using customer-managed keys with Autonomous Data Guard
  - Autonomous Data Guard with Customer Managed Keys
  - Notes for Customer-Managed Keys with Autonomous Data Guard

- When you enable Autonomous Data Guard with a cross-region standby database, the wallets for the primary and the standby specify different database hostnames and use different connection strings. Oracle recommends that applications use the connection string or wallet downloaded from the same region as the primary database.

  If you need to use a single connection string or wallet containing both the primary and the standby database hostnames, you may construct this manually.

  To manually construct a wallet that contains both the primary and the remote database connections strings:

  1. From the primary database's Oracle Cloud Infrastructure Console, click **Database connection** to download the primary's `wallet.zip`.
  2. From the remote standby database's Oracle Cloud Infrastructure Console, click **Database connection** to download the standby's `wallet.zip`.
  3. Unzip both wallet files and open the two `tnsnames.ora` files.
  4. Copy the remote database's connect descriptor into the primary database's connection string in the primary's `tnsnames.ora` file using your preferred retry delays.
  5. Zip the updated primary database wallet folder.

  With this updated `tnsnames.ora`, your primary database connection strings in the updated `wallet.zip` will contain both the primary and the standby hostnames, to support failover. An application using the updated wallet attempts to connect and retries connecting to the first listed database hostname, and if that connection fails due to the database being Unavailable, the application then automatically attempts to connect to the second database hostname.

  For example, if your Autonomous Data Guard is setup with the primary in Ashburn (IAD) and a cross-region standby in Phoenix (PHX), Oracle recommends that your mid-tier applications running in IAD use the connection string or wallet from that of the primary database in IAD, and your corresponding applications running in PHX use the connection string or wallet from that of the standby database in PHX. For a regional failover or switchover, Oracle recommends failing over both your database and your application or mid-tier, for optimum performance and to minimize any cross-regional latency.

  For example:

```
a6gxf2example9ep_high = (description_list=
     (failover=on) (load_balance=off)
     (description= (retry_count=15)(retry_delay=3)(address=(protocol=tcps)
(port=1522)(host=adb.us-ashburn-1.oraclecloud.com))
(connect_data=(service_name=mqssyowmexample_a6gxf2example9ep_high.adb.oracl
ecloud.com))(security=(ssl_server_dn_match=yes)))
     (description= (retry_count=15)(retry_delay=3)(address=(protocol=tcps)
(port=1522)(host=adb.us-phoenix-1.oraclecloud.com))
(connect_data=(service_name=mqssyowmexample_a6gxf2example9ep_high.adb.oracl
ecloud.com))(security=(ssl_server_dn_match=yes))))

a6gxf2example9ep_low =
     (description_list= (failover=on) (load_balance=off)
     (description= (retry_count=15)(retry_delay=3)(address=(protocol=tcps)
(port=1522)(host=adb.us-ashburn-1.oraclecloud.com))
(connect_data=(service_name=mqssyowmexample_a6gxf2example9ep_low.adb.oracle
cloud.com))(security=(ssl_server_dn_match=yes)))
     (description= (retry_count=15)(retry_delay=3)(address=(protocol=tcps)
(port=1522)(host=adb.us-phoenix-1.oraclecloud.com))
(connect_data=(service_name=mqssyowmexample_a6gxf2example9ep_low.adb.oracle
```

```
cloud.com))(security=(ssl_server_dn_match=yes))))

a6gxf2example9ep_medium =
     (description_list= (failover=on) (load_balance=off)
     (description= (retry_count=15)(retry_delay=3)(address=(protocol=tcps)
(port=1522)(host=adb.us-ashburn-1.oraclecloud.com))
(connect_data=(service_name=mqssyowmexample_a6gxf2example9ep_medium.adb.ora
clecloud.com))(security=(ssl_server_dn_match=yes)))
     (description= (retry_count=15)(retry_delay=3)(address=(protocol=tcps)
(port=1522)(host=adb.us-phoenix-1.oraclecloud.com))
(connect_data=(service_name=mqssyowmexample_a6gxf2example9ep_medium.adb.ora
clecloud.com))(security=(ssl_server_dn_match=yes))))
```

Autonomous Database Cross-Region Paired Regions

Autonomous Database provides paired regions for creating cross-region databases, including cross-region clones and cross-region disaster recovery peers.
The **Region** list is shows paired regions when you add a remote disaster recovery peer or when you create a clone in a different region. The **Region** list only shows paired regions. That is, if your tenancy is not subscribed to region that is paired to the source region, then the region is not displayed. To subscribe to a region, see Managing Regions.

If you do not see a region that you are subscribed to in the **Region** list, open a support ticket to request that the region be added as a paired region for your source region. See Get Help, Search Forums, and Contact Support for more information.

Backup and Restore Autonomous Database Instances

This section describes backup and recovery tasks on Autonomous Database.

- About Backup and Recovery on Autonomous Database
- Edit Automatic Backup Retention Period on Autonomous Database
- Create Long-Term Backups on Autonomous Database
- View Backup Information and Backups on Autonomous Database
  You can view backup information and view the available backups on Autonomous Database.
- Restore and Recover your Autonomous Database
  From the Oracle Cloud Infrastructure Console you can restore the database using the **Restore** operation, where you initiate recovery for your database.
- Backup and Restore Notes
  Provides notes for automatic backups, long-term backups, and restoring your database.

About Backup and Recovery on Autonomous Database

Describes Autonomous Database backup options.

All backups taken and managed by Oracle in Autonomous Database are immutable. Immutable backups cannot be deleted or modified.

Automatic Backups

Autonomous Database performs periodic backups including full, cumulative, and incremental backups, to ensure data reliability and recoverability.

The choices for automatic backups are different depending on your compute model:

- **OCPU compute model**
  Autonomous Database automatically backs up your database for you. The retention period for backups is 60 days. You can restore and recover your database to any point-in-time in this retention period.
- **ECPU compute model**
  By default Autonomous Database automatically backs up your database for you and the retention period for backups is 60 days. Optionally, you can choose the backup retention period in days, from 1 to 60. You can restore and recover your database to any point-in-time in the specified retention period.

Billing for automatic backups depends on the compute model that you select when you provision or clone an Autonomous Database instance:

- **OCPU compute model**: the storage for automatic backups is included in the cost of database storage.
- **ECPU compute model**: the storage for backups is billed separately and in addition to database storage.

See How Is Autonomous Database Billed? for more information.

Recovery

You can initiate recovery for your Autonomous Database using the Oracle Cloud Infrastructure Console. Autonomous Database automatically restores and recovers your database to the point-in-time you specify or using the backup you select from a list of backups.

See Restore and Recover your Autonomous Database for more information.

Clone from a Backup

You can use either an automatic backup or a long-term backup to create a clone of your database.

See Clone an Autonomous Database from a Backup for more information.

Listing Backups

The list of backups available for recovery is shown on the Autonomous Database details page under **Backups**. Click **Backups** under **Resources** to show the backups.

The Oracle Cloud Infrastructure Console Autonomous Database details page, in the Backup area shows the **Total backup storage** field. This fields shows the total storage being billed, including for automatic backups and if there are long-term backups, this also includes the long term backup storage.

See View Backup Information and Backups on Autonomous Database for more information.

- Long-Term Backups on Autonomous Database
  Using the long-term backup feature you can create a long-term backup as a one-time backup or as scheduled long-term backup. You select the retention period for long-term backups in the range of a minimum of 3 months and a maximum of 10 years.

Long-Term Backups on Autonomous Database

Using the long-term backup feature you can create a long-term backup as a one-time backup or as scheduled long-term backup. You select the retention period for long-term backups in the range of a minimum of 3 months and a maximum of 10 years.

Autonomous Database takes scheduled long-term backups automatically according to the schedule you create: one-time, weekly, monthly, or yearly.

You can create a new database by cloning from a long-term backup. You cannot use a long-term backup to recover or restore the same database where the long-term backup was created.

Long-term backup billing:

- **ECPU compute model:** Long-term backup storage is billed in addition to your database storage. The total of your automatic backups and long-term backups will be billed additionally as backup storage.

  See ECPU Compute Model Billing Information for more information about how backup storage is billed.

- **OCPU compute model**: Using the OCPU compute model, long-term backups are billed in addition to your database storage and long term backup storage is billed at the same rate as database storage. Long-term backup storage is billed based on the storage usage, rounded to the nearest Terabyte. If your long-term backup storage usage is 0.5 TB, you are billed for 1 TB (at the same rate as database storage).

  For example, if you provision your Autonomous Database with 3 TB of storage and you create several long-term backups amounting to 1.2 TB, you are billed for a total of 5 TB of storage, as follows:

  – 3 TB (the provisioned storage)

    plus

  – 2 TB (1.2TB of long-term backups rounded up)

  See OCPU Compute Model Billing Information for more information about how backup storage is billed.

Each long term backup is a standalone backup that allows you to create a point-in-time clone, based on the long-term backup timestamp.

Edit Automatic Backup Retention Period on Autonomous Database

You can change the backup retention period for automatic backups on Autonomous Database with a retention period between 1 day and up to 60 days.

Editing the backup retention period is only available for instances using the ECPU compute model. See Compute Models in Autonomous Database for more information.

> **Note:**
>
> The storage for backups incurs additional costs. See ECPU Compute Model Billing Information for more information.

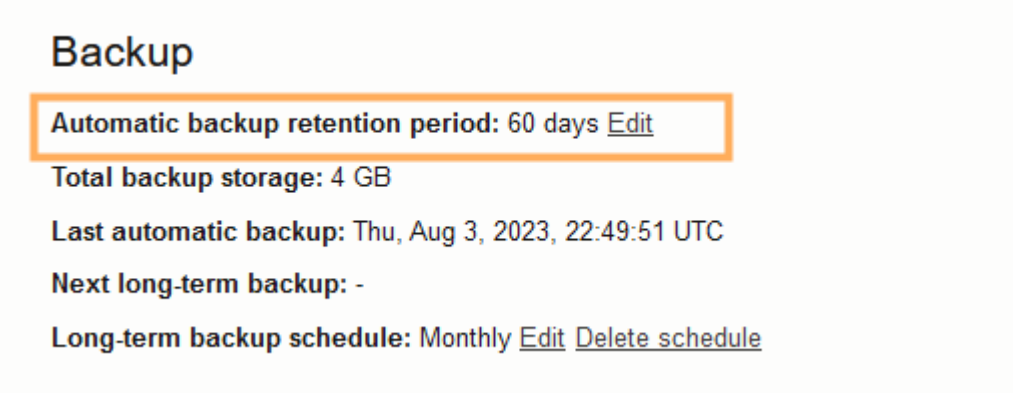
Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.

- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

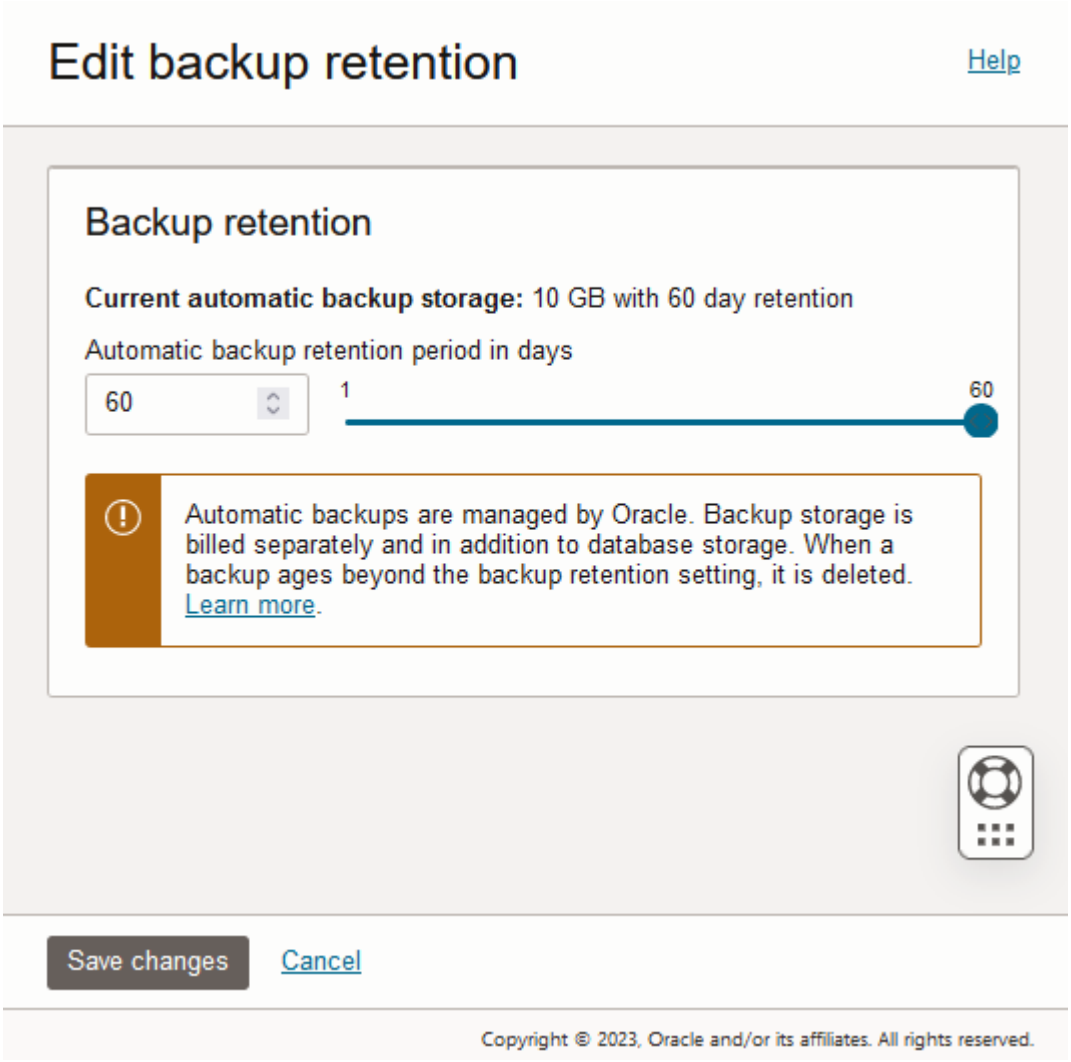
To change the automatic backup retention period:

1. On the Autonomous Database Details page, under **Backup**, in the **Automatic backup retention period** field, click **Edit**.



   displays the Edit backup retention dialog.

2. To edit the automatic backup retention period, enter the automatic backup retention period (in days).
   - Enter a value in the text field or use the arrows to increase or decrease the backup retention period.
   - Use the slider to select the backup retention period.



3. Click **Save changes**.

Notes for specifying the automatic backup retention period:

- The backup retention period specifies how many days automatic backups are retained and the period for which you are billed for the automatic backups that are retained. If you change the backup retention period to a shorter period and automatic backups exist that are older, you are no longer billed for the automatic backups with timestamps beyond the specified retention period (and Autonomous Database may delete the older automatic backups).

  If you change the backup retention period to a longer period, for example if you change from 7 to 60 days, Autonomous Database starts retaining automatic backups up to 60 days, and billing for the retained backups. If older automatic backups already exist and were not deleted when you changed to a shorter backup retention period, the backups that have not been deleted show up and you are again billed for the automatic backups.

- The backup retention period for a remote peer database follows that of the source Primary database. Automatic backups are only taken on a peer database when it becomes the Primary database after a switchover or after a failover.

  See Use Standby Databases with Autonomous Data Guard for Disaster Recovery and Use Backup-Based Disaster Recovery for more information.

Create Long-Term Backups on Autonomous Database

You can create long-term backups on Autonomous Database with a retention period between three (3) months and up to ten (10) years.

When you create a long-term backup you select to create a one-time backup or set a schedule to automatically create backups that are taken weekly, monthly, or annually (yearly).

You can use a long-term backup to clone a new database. You may clone a new database from any timestamp between a long-term backup timestamp and up to 24 hours post (after) its timestamp. See Clone an Autonomous Database from a Backup for more information.

> **Note:**
>
> Long-term backups incur additional costs. See Long-Term Backups on Autonomous Database for more information.

The Autonomous Database instance must have at least one automatic backup before you can create a long term backup. After you provision or clone an Autonomous Database instance, you may need to wait up to 4 hours before an automatic backup is available. See View Backup Information and Backups on Autonomous Database to find out if a backup exists.

Perform the following prerequisite steps as necessary:

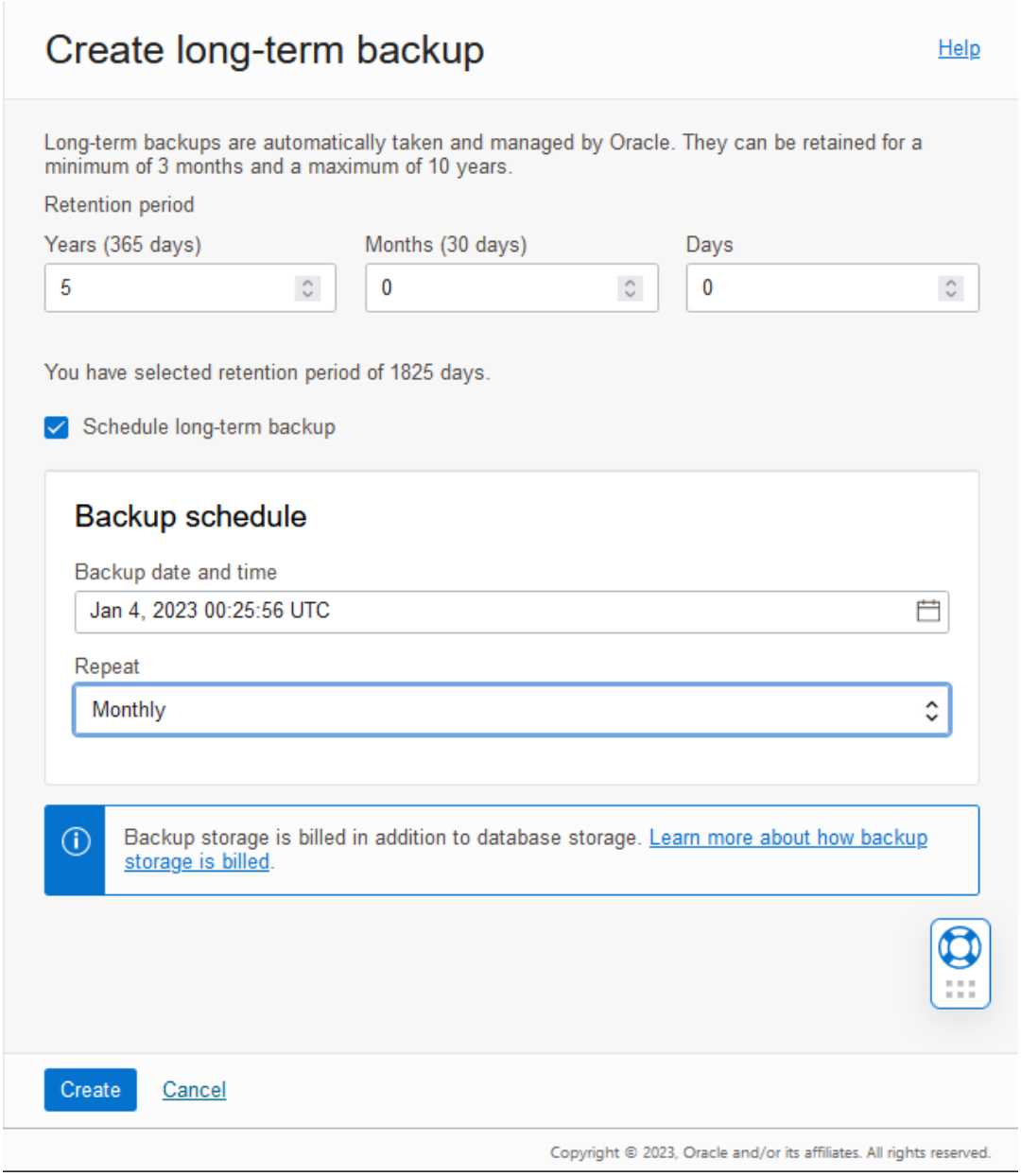
- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To create a long-term backup:

1. On the Autonomous Database Details page, under **Resources**, click **Backups**.
2. On the Autonomous Database Details page, under **Backups**, click **Create long-term backup**.
3. Set the long-term backup retention period.

   In the **Years**, **Months**, and **Days** fields, accept the default value, enter a value, or increment the values to increase or decrease the retention period for each of these choices.
4. If you want to begin the backup immediately, do not select **Schedule long-term backup** and skip to the next step.

   If you want to create a backup schedule, select **Schedule long-term backup**.



a. Enter a **Backup date and time**.

b. Select a repeat interval from the **Repeat** list.

When you select **One-time (does not repeat)**, this schedules a single long-term backup to be created on the specified date and time.

Selecting either **Weekly** specifies that Autonomous Database repeatedly creates long-term backups at the specified time at a weekly, every 7 days, interval. That is, backups are created and repeat at the same time weekly. For example, if the Backup date and Time is Jan 24, 2023 00:09:28 UTC and this is a Tuesday, and **Weekly** is selected, the long-term backup will happen every Tuesday.

Selecting **Monthly** specifies that Autonomous Database repeatedly creates long-term backups at the specified time each month. When you select **Monthly**, the backup will occur every month at the date you select. If you select a day of the month >= 29, Autonomous Database takes the backup on the last the last day of the month instead (to account for months with fewer days and leap years).

Selecting **Yearly** specifies that Autonomous Database repeatedly creates long-term backups at the specified time each year. When you select **Yearly**, the backup will occur every year at the date you select. If you select a day of the month >= 29, Autonomous Database takes the backup on the last the last day of the month instead (to account for months with fewer days and leap years).

5. Click **Create**.

   If you did not select **Schedule long-term backup**, Autonomous Database begins creating a long-term backup immediately.

- Edit Long-Term Backup Schedule
  You can edit the long-term backup schedule for an Autonomous Database instance.
- Delete Long-Term Backup Schedule
  You can delete the long-term backup schedule for an Autonomous Database instance.
- Delete a Long-Term Backup
  You can delete a long-term backup.

Edit Long-Term Backup Schedule

You can edit the long-term backup schedule for an Autonomous Database instance.

1. On the Autonomous Database Details page, in the **Long-term backup schedule** field under Backup, click **Edit**.

   This shows the Configure long-term backup schedule dialog.
2. Make changes to the long-term backup schedule.
3. Click **Save**.

Delete Long-Term Backup Schedule

You can delete the long-term backup schedule for an Autonomous Database instance.

1. On the Autonomous Database Details page, in the **Long-term backup schedule** field under Backup, click **Delete Schedule**.
2. In the Delete Schedule confirmation dialog, click **Delete**.

Delete a Long-Term Backup

You can delete a long-term backup.

To delete a long-term backup,

1. On the Autonomous Databases page select your Autonomous Database from the links under the **Display Name** column.
2. On the Autonomous Database Details page, under **Resources**, click **Backups**.
3. In the table, from the list of backups select the long-term backup you want to delete and click  at the end of a row.
4. In the list, select **Delete**.

View Backup Information and Backups on Autonomous Database

You can view backup information and view the available backups on Autonomous Database.

The Oracle Cloud Infrastructure Console shows information about backups. To view general backup information, view the Backup area on the Autonomous Database details page. Depending on your compute model, the Backup area includes the following information:

- **Automatic backup retention period**:
- **Total backup storage**: Shows the total storage being billed, including for automatic backups and if there are long-term backups, this also includes the long term backup storage.
- **Last automatic backup**: Shows the timestamp of the last automatic backup.
- **Next long-term backup**: When you configure scheduled long-term backups, this shows the timestamp for the next long-term backup.
- **Long-term backup schedule**: Shows the long-term backup schedule, with links to edit or delete a long-term backup schedule.

To list completed backups:

1. On the Autonomous Databases page select your Autonomous Database from the links under the **Display Name** column.
2. On the Autonomous Database Details page, under **Resources**, click **Backups**.

This shows the list of backups for the Autonomous Database instance.

If you click  at the end of a row for a backup in the Backups list, this shows the actions you can perform.

- **Actions for Automatic Backups**:
  - Restore: See Restore and Recover your Autonomous Database for more information.
  - Create clone: See Clone an Autonomous Database from a Backup for more information.
- **Actions for long-term backups**:
  - Create clone: Clone an Autonomous Database from a Backup for more information.
  - Edit retention period: This allows you to change the retention period for a long-term backup.
  - Delete: This allows you to remove a long-term backup.

Restore and Recover your Autonomous Database

From the Oracle Cloud Infrastructure Console you can restore the database using the **Restore** operation, where you initiate recovery for your database.

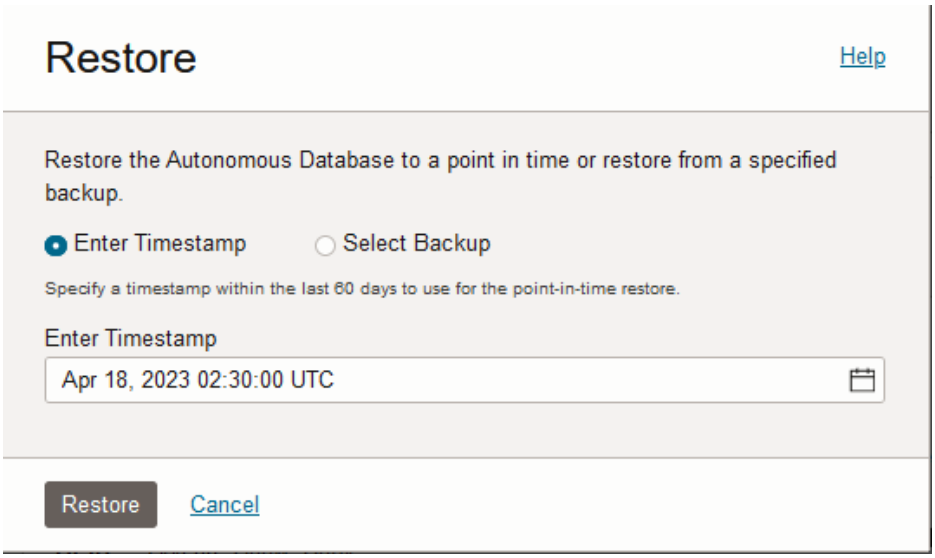
Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.

- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To restore and recover your database, do the following:

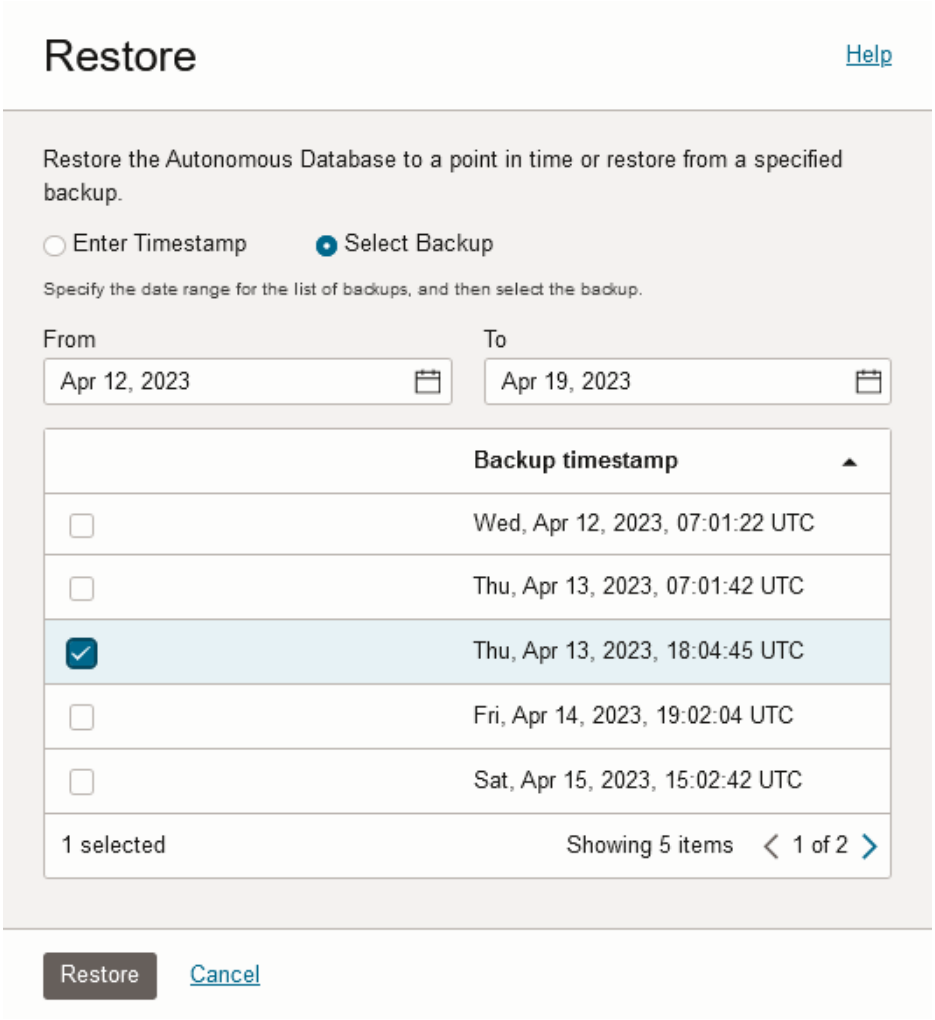
1. On the Autonomous Database Details page, from the **More actions** drop-down list, select **Restore** to display the **Restore** prompt.
2. In the Restore prompt, select **Enter Timestamp** or **Select Backup** to restore to a point in time or to restore from a specified backup.
   - **Enter Timestamp**: Enter a timestamp to restore to in the **Enter Timestamp** calendar field.



     Click the Calendar icon to show the date and timestamp calendar selector. If automatic backups are enabled on your instance, you can select a timestamp to restore and recover your database to any point-in-time in the retention period.

     Manually edit the values in the **Enter Timestamp** field if you prefer to use a more granular timestamp.

   - **Select Backup**: Select a backup from the list of backups. Limit the number of backups you see by specifying a period using the **From** and **To** calendar fields.



3. Click **Restore**.

> **Note:**
>
> Restoring Autonomous Database puts the database in the unavailable state during the restore operation. You cannot connect to a database in that state. The only lifecycle management operation supported in unavailable state is **Terminate**.

When concurrent operations such as scaling are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

The details page shows Lifecycle State: **Restore In Progress**

4. When the restore operation finishes your Autonomous Database instance opens and the Lifecycle State shows **Available**.

At this point you can connect to your Autonomous Database instance and check your data to validate that the restore point you specified was correct. After checking your data if you find that the restore date you specified was not the one you needed, you can initiate another restore operation to another point in time.

Notes:

- The restore operation also restores the `DATA_PUMP_DIR` directory and user defined directories to the timestamp you specified for the restore; files that were created after that timestamp would be lost.
- When you restore, the Oracle Machine Learning workspaces, projects, and notebooks are not restored.
- For external tables, partitioned external tables, and the external partitions of hybrid partitioned tables a backup does not include the external files that reside on your Object Store. Thus, for operations where you use a backup to restore your database, such as **Restore** or **Clone from a backup** it is your responsibility to backup, and restore if necessary, the external files associated with external tables, external partitioned tables, or the external files for a hybrid partitioned table.

  See Clone an Autonomous Database from a Backup for information on using **Clone from a backup**.

Backup and Restore Notes

Provides notes for automatic backups, long-term backups, and restoring your database.

- **Files on Object Store**: For external tables, partitioned external tables, and the external partitions of hybrid partitioned tables, backups do not include the external files that reside on Object Store. Thus, for operations where you use a backup to restore your database, such as **Restore** or **Clone from a backup**, it is your responsibility to backup and restore if necessary, the external files associated with external tables, external partitioned tables, or the external files for a hybrid partitioned table.

  See Restore and Recover your Autonomous Database for information on **Restore**.

  See Clone an Autonomous Database from a Backup for information on **Clone from a backup**.
- **Stopped Database**: Automatic backups occur when a database is stopped. If you stop your database, you do not miss a regularly scheduled automatic backup.
- **Manual Backups**: The Oracle Cloud Infrastructure Console does not provide the option to create a manual backup. Oracle recommends that you do not create or use manual backups. If required, manual backups are available using the API. See CreateAutonomousDatabaseBackup for more information.

Long-Term Backup Notes

- Long-term backups are retained and managed by Oracle for the specified retention period you select when you create or schedule long-term backups.
- Each long-term backup creates a full backup and the backup can only be used from the Autonomous Database instance to clone a new database.
- After the long-term retention period is over a backup will be deleted and is no longer retained.
- The encryption key option that is selected when a long-term backup is created applies when you create a clone from the long-term backup. If a customer-managed master encryption key is in use at the time a long-term backup is created, then the same

customer-managed master encryption key must be available to access a database that you clone from the long-term backup.

See Manage Encryption Keys on Autonomous Database for more information.

- While backing up a database, the database is fully functional. However, during a long-term backup certain lifecycle management operations, such as stopping the database may affect the backup. See Long-Term Backup with Concurrent Operations for more information.
- Cloning from a long-term backup creates a new database with the most current available database version, which is possibly not the original database version in use at the time when the long-term backup was created. A database cloned from a long-term backup uses latest available database version.

  For example, if a long-term backup is taken on a database with version 19c, and then 4 years later the long-term backup is used to clone a new database, the new database may only offer the latest database version (for example Oracle Database 23ai, if version 19c is not available).
- Long-term backups are only accessible while a database exists. If you terminate an Autonomous Database instance, you no longer have access to long-term backups. If you want to retain long-term backups and minimize costs:

  1. Drop all tables and scale down the database to the minimum storage.
  2. Stop the database.

  This preserves the long-term backups that were created for the Autonomous Database instance.

Use OraMTS Recovery Feature on Autonomous Database

Use Oracle MTS (OraMTS) Recovery Service to resolve an in-doubt Microsoft Transaction Server transaction.

- About OraMTS Recovery Service
  The Oracle MTS (OraMTS) Recovery Service resolves an in-doubt Microsoft Transaction Server transaction.
- Prerequisites to Enable OraMTS Recovery Service on Autonomous Database
  Lists the prerequisites to enable OraMTS Recovery Service for an Autonomous Database.
- Enable OraMTS Recovery Service on an Autonomous Database
  Describes steps to enable OraMTS Recovery Service on an Autonomous Database.
- Disable OraMTS Recovery Service on an Autonomous Database
  Describes steps to disable OraMTS Recovery Service for an Autonomous Database.

About OraMTS Recovery Service

The Oracle MTS (OraMTS) Recovery Service resolves an in-doubt Microsoft Transaction Server transaction.

An Microsoft Transaction Server (MTS) is a COM-based transaction processing system that runs on an internet or network server.

An Oracle MTS Recovery Service is automatically installed with Oracle Services For Microsoft Transaction Server. The Oracle MTS (OraMTS) Recovery Service resolves in-doubt transactions on the computer that started the failed transaction. A scheduled recovery job for each MTS-enabled database lets the OraMTS Recovery Service resolve in-doubt transactions.

To use MTS with an Oracle database, distributed transaction capabilities are necessary.

When any of these components fails, Oracle transactions connected to Microsoft Transaction Server become in-doubt transactions:

- Microsoft Transaction Server application
- Network
- Microsoft Distributed Transaction Coordinator (MS DTC)

See Using Microsoft Transaction Server with Oracle Database for more information.

The Oracle MTS (OraMTS) Recovery Service resolves an in-doubt Microsoft Transaction Server transaction in this order:

1. The DBMS recovery job detects an in-doubt MTS-related transaction.
2. The DBMS recovery job extracts the recovery service's endpoint address from the `XID` of the in-doubt transaction and requests the recovery service for the outcome of the MTS/MS DTC transaction.
3. The recovery service requests its MS DTC for transaction outcome.
4. The recovery service reports transaction outcome to the DBMS job process.
5. The DBMS recovery job commits or terminates the in-doubt transaction based on the outcome reported by MS DTC.

> **Note:**
>
> Each computer can only have one instance of Oracle MTS (OraMTS) Recovery Service installed.

Prerequisites to Enable OraMTS Recovery Service on Autonomous Database

Lists the prerequisites to enable OraMTS Recovery Service for an Autonomous Database.

To enable Oracle MTS Recovery Service on an Autonomous Database:

- You must configure your database on a private endpoint.
- For your OraMTS Recovery Service, you must deploy the VM in the same private network as the database.
- You must configure an OCI Private Load Balancer (LBaaS) and the Load Balancer (LBaaS) must be able to access the VM on port 2030. See Load Balancer Management for more information.
- Your database must be able to communicate with the Load Balancer (LBaaS) on port 443. To enable this you need an egress rule for port 443 in VCN's security list or in the network security group.
- Your Load Balancer (LBaaS) must also be able to receive the communication from the database. To enable this you need an ingress rule for your Load Balancer (LBaaS) for port 443.
- Reserve a domain name with a domain provider.
- Generate an SSL certificate for the domain.
- You must configure a secure HTTPS endpoint using OCI Load Balancer to ensure that the communication between the Autonomous Database and the MTS server uses HTTPS

protocol with SSL encryption. See Configure Network Access with Private Endpoints and Submit an HTTP Request to a Private Host with UTL_HTTP for more information.

Enable OraMTS Recovery Service on an Autonomous Database

Describes steps to enable OraMTS Recovery Service on an Autonomous Database.

To enable OraMTS Recovery Service on your Autonomous Database, you must be logged in as the `ADMIN` user or have the `EXECUTE` object privilege on `DBMS_CLOUD_ADMIN`.

Run `DBMS_CLOUD_ADMIN.ENABLE_FEATURE` to enable OraMTS Recovery Service on your Autonomous Database.

Example to Enable and Verify the OraMTS Recovery Service:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_FEATURE(
        feature_name => 'ORAMTS',
        params       => JSON_OBJECT('location_uri' VALUE 'https://
mymtsserver.mycorp.com')
   );
END;
/


SELECT property_value FROM database_properties WHERE property_name =
'ORAMTS_SERVER';
```

The first example enables the OraMTS Recovery Service on your Autonomous Database.

The `feature_name` parameter specifies the name of the feature to enable. The `ORAMTS` value indicates that you are enabling the OraMTS recovery service feature for your database.

The `location_uri` parameter specifies the HTTPS URL for the OraMTS server in a customer network.

The second example is a SQL statement that you can run to verify that the OraMTS Recovery Service is enabled for your Autonomous Database.

See ENABLE_FEATURE Procedure for more information.

Disable OraMTS Recovery Service on an Autonomous Database

Describes steps to disable OraMTS Recovery Service for an Autonomous Database.

To disable OraMTS Recovery Service on your Autonomous Database, you must be logged in as the `ADMIN` user or have the `EXECUTE` object privilege on `DBMS_CLOUD_ADMIN`.

Run `DBMS_CLOUD_ADMIN.DISABLE_FEATURE` to disable OraMTS Recovery Service on your Autonomous Database.

Example to disable the OraMTS Recovery Service:

```
BEGIN
DBMS_CLOUD_ADMIN.DISABLE_FEATURE(
    feature_name => 'ORAMTS');
```

```
END;
/
```

This disables the OraMTS Recovery feature on your Autonomous Database.

The `feature_name` parameter specifies the name of the feature to disable. The `ORAMTS` value indicates that you are disabling the OraMTS recovery service feature for your database.

See DISABLE_FEATURE Procedure for more information.

Use Backup-Based Disaster Recovery

Backup-Based Disaster Recovery provides a lower-cost disaster recovery option for Autonomous Database (the RTO is higher with this option, as compared to using Autonomous Data Guard).

Autonomous Database Disaster Recovery Options

Backup-Based Disaster Recovery uses backups to instantiate a peer database at the time of switchover or failover. For local backup-based disaster recovery, existing local backups are utilized. There are no additional costs for a local Backup-Based Disaster Recovery. Cross-Region Backup-Based Disaster Recovery incurs an additional cost.

Backup-Based Disaster Recovery is available for all workload types.

Autonomous Data Guard When you add an Autonomous Data Guard standby database, the system creates a standby database that continuously gets updated with the changes from the primary database. You can use Autonomous Data Guard with a standby in the current region, a local standby, or with a standby in a different region, a cross-region standby. You can also use Autonomous Data Guard with both a local standby and a cross-region standby. See Use Standby Databases with Autonomous Data Guard for Disaster Recovery for information about Autonomous Data Guard.

Autonomous Data Guard is available for the following workload types: Data Warehouse and Transaction Processing.

- About Backup-Based Disaster Recovery
  Backup-Based Disaster Recovery uses backups to instantiate a peer database at the time of switchover or failover. This enables you to have a lower cost and higher Recovery Time Objective (RTO) disaster recovery option for your Autonomous Database, as compared with Autonomous Data Guard.
- View Backup-Based Disaster Recovery Peer
  Backup-Based Disaster Recovery with a local peer is enabled by default for a newly created Autonomous Database instance, and for existing databases. A local Backup-Based Disaster Recovery peer does not incur any additional cost.
- Add a Cross-Region Disaster Recovery Peer
  In addition to a local Backup-Based Disaster Recovery peer, you can add a second peer that is a remote (cross-region) Backup-Based Disaster Recovery peer.
- Update Disaster Recovery Type
  Describes the steps to change to an alternative Disaster Recovery option.
- Disable a Cross-Region (Remote) Peer
  Describes the steps to terminate a cross-region (remote) peer.
- Perform a Switchover to a Backup Copy Peer
- Perform a Failover

- Convert Cross Region Backup-Based Disaster Recovery Peer to Snapshot Standby
- Update Remote Peer Network ACLs for a Backup-Based Disaster Recovery Peer
- Use the API
  Provides links for details on using API operations to manage Backup-Based Disaster Recovery.
- Backup-Based Disaster Recovery Events
  You can use Oracle Cloud Infrastructure events to respond when Autonomous Database changes its state due to a Backup-Based Disaster Recovery related event.

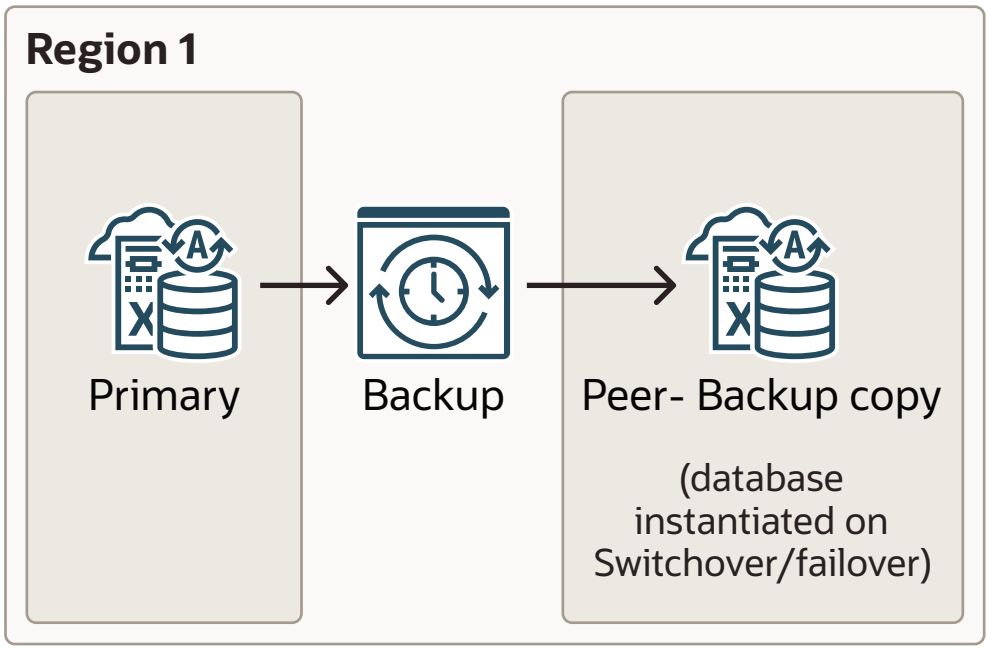
About Backup-Based Disaster Recovery

Backup-Based Disaster Recovery uses backups to instantiate a peer database at the time of switchover or failover. This enables you to have a lower cost and higher Recovery Time Objective (RTO) disaster recovery option for your Autonomous Database, as compared with Autonomous Data Guard.

> **Note:**
>
> Backup-Based Disaster Recovery (backup copy) is available in all Autonomous Database workload types. Backup-Based Disaster Recovery is not available with Always Free Autonomous Database.

Backup-Based Disaster Recovery with Local Peer

For local backup-based disaster recovery, existing local backups are utilized. There are no additional costs for local Backup-Based Disaster Recovery.



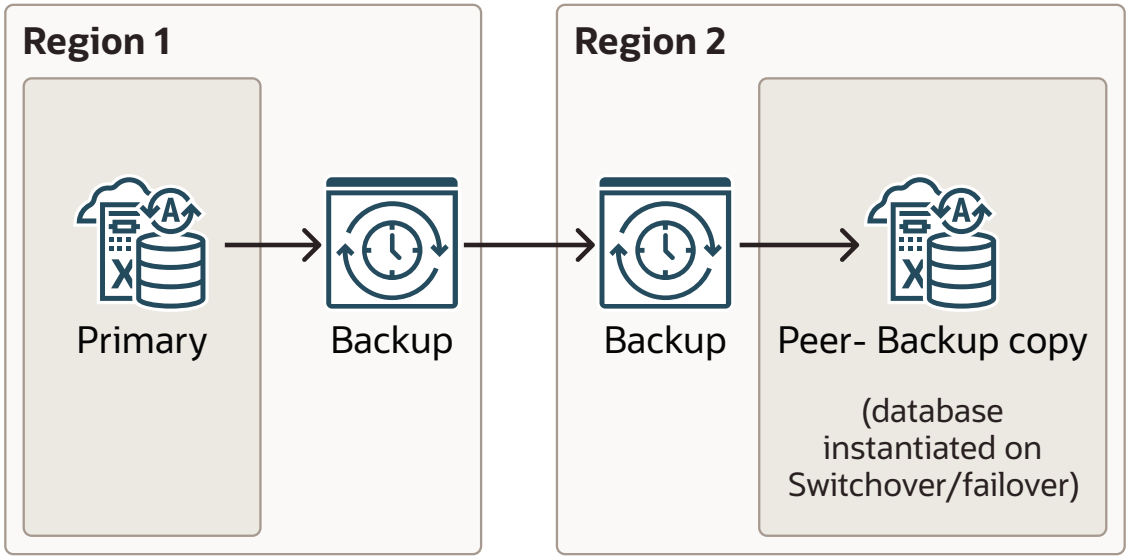
For better resilience for Backup-Based Disaster Recovery, a peer is instantiated as follows:

- In regions with more than one availability domain, a local peer is instantiated in a different availability domain than the primary database.
- In regions with a single availability domain, a local peer is instantiated in a different fault domain than the primary database (that is, on a different physical machine).

All Autonomous Database features from the primary database are available when a peer is instantiated and becomes the primary, after the system fails over or after you perform a switchover operation. See Autonomous Data Guard with Local Standby for more information.

Backup-Based Disaster Recovery with Cross-Region Peer

For backup-based disaster recovery with a cross-region peer, backups are copied to the remote region. Cross-Region Backup-Based Disaster Recovery incurs additional costs.



Topics

- Backup-Based Disaster Recovery Recovery Time Objective (RTO) and Recovery Point Objective (RPO)
  When you perform a failover using with Backup-Based Disaster Recovery enabled, the peer instance assumes the role of the primary instance, according to the Recovery Time Objective (RTO) and Recovery Point Objective (RPO).
- Replicating Backups to a Cross-Region Backup Based Disaster Recovery Peer
  When you add a cross-region Backup-Based Disaster Recovery peer you can enable cross-region backup replication for automatic backups.

Backup-Based Disaster Recovery Recovery Time Objective (RTO) and Recovery Point Objective (RPO)

When you perform a failover using with Backup-Based Disaster Recovery enabled, the peer instance assumes the role of the primary instance, according to the Recovery Time Objective (RTO) and Recovery Point Objective (RPO).

The RTO is the maximum amount of time required to restore database connectivity to a backup copy database after a failover is initiated. The RPO is the maximum duration of potential data loss, in minutes, on the primary database.

Backup-Based Disaster Recovery RTO and RPO numbers are:

Backup-Based Disaster Recovery Configuration	RTO	RPO
Local backup copy	one (1) hour + 1 hour per 5 TB	10 seconds

Backup-Based Disaster Recovery Configuration	RTO	RPO
Cross-region (remote) backup copy	one (1) hour + 1 hour per 5 TB	1 min

Replicating Backups to a Cross-Region Backup Based Disaster Recovery Peer

When you add a cross-region Backup-Based Disaster Recovery peer you can enable cross-region backup replication for automatic backups.

By default, automatic backups are created and maintained at the current Primary database and are not replicated to a cross-region peer. Optionally, you can enable replication of the automatic backups to the cross region peer.

When you enable cross-region backup replication, up to 7 days of automatic backups for the primary are replicated to a cross-region peer. When this feature is enabled, automatic backups are available in the remote region as follows:

- After a switchover or failover you can restore or clone to any timestamp in the past seven (7) days, or to any timestamp in the specified retention period when the retention period is set to less than seven days.
- All backups for the primary that are replicated to the remote region are deleted on the remote region peer after seven days, or after the retention period number of days when the retention period is set to less than seven days.
- You cannot modify the backup retention period for replicated backups, except if you modify the backup retention period on the primary to specify a value less than seven days. In this case, the retention period for replicated backups on the remote region matches the automatic backup retention period set on the primary.

The cross-region backup replication incurs an additional cost. See Oracle Autonomous Database Serverless Features Billing for more information.

See the following for more information:

- Add a Cross-Region Disaster Recovery Peer
- Enable or Disable Backup Replication for an Existing Cross Region Peer

Note the following for cross-region backup replication:

- After a switchover or a failover, while the cross-region database is in the primary role, backups are taken on the current primary and are replicated to the current (remote) peer.
- When using Backup-Based Disaster Recovery with a cross-region peer, this feature is supported for all workload types.

View Backup-Based Disaster Recovery Peer

Backup-Based Disaster Recovery with a local peer is enabled by default for a newly created Autonomous Database instance, and for existing databases. A local Backup-Based Disaster Recovery peer does not incur any additional cost.

Perform the following prerequisite steps as necessary:

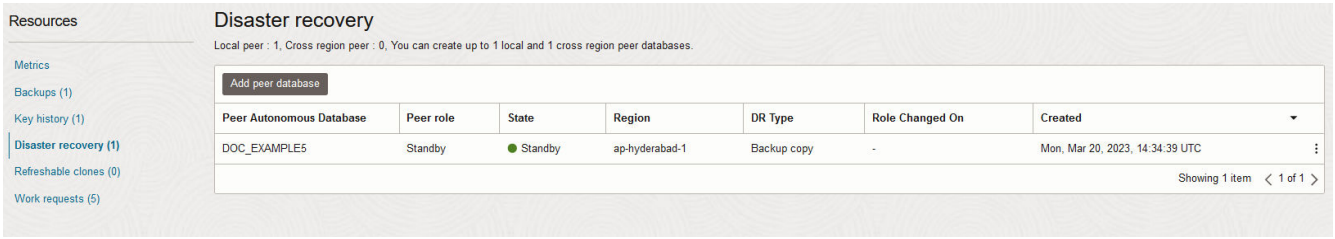
- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.

- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To view the current Disaster recovery configuration for your Autonomous Database, do the following:

On the Autonomous Database Details page, in the **Resources** area click **Disaster recovery**. The **DR Type** field indicates the type of disaster recovery, either Backup-Based Disaster Recovery (Backup copy) or Autonomous Data Guard.

For example:



Add a Cross-Region Disaster Recovery Peer

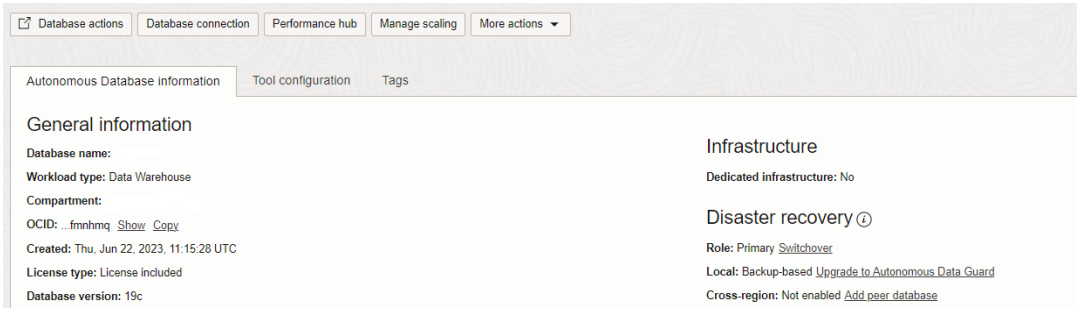
In addition to a local Backup-Based Disaster Recovery peer, you can add a second peer that is a remote (cross-region) Backup-Based Disaster Recovery peer.

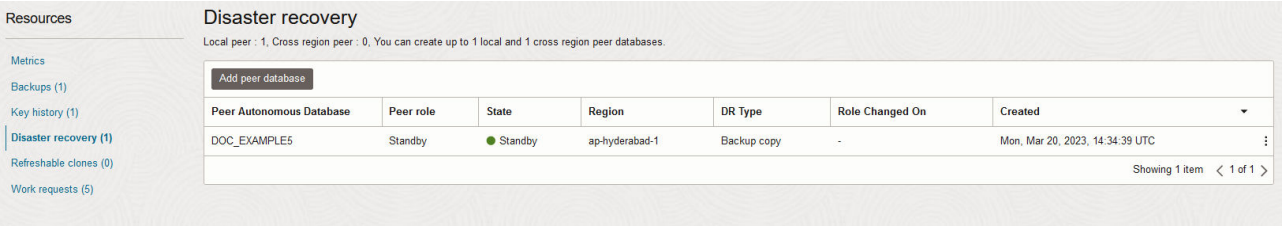
Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To add a cross-region Backup-Based Disaster Recovery peer, do the following:

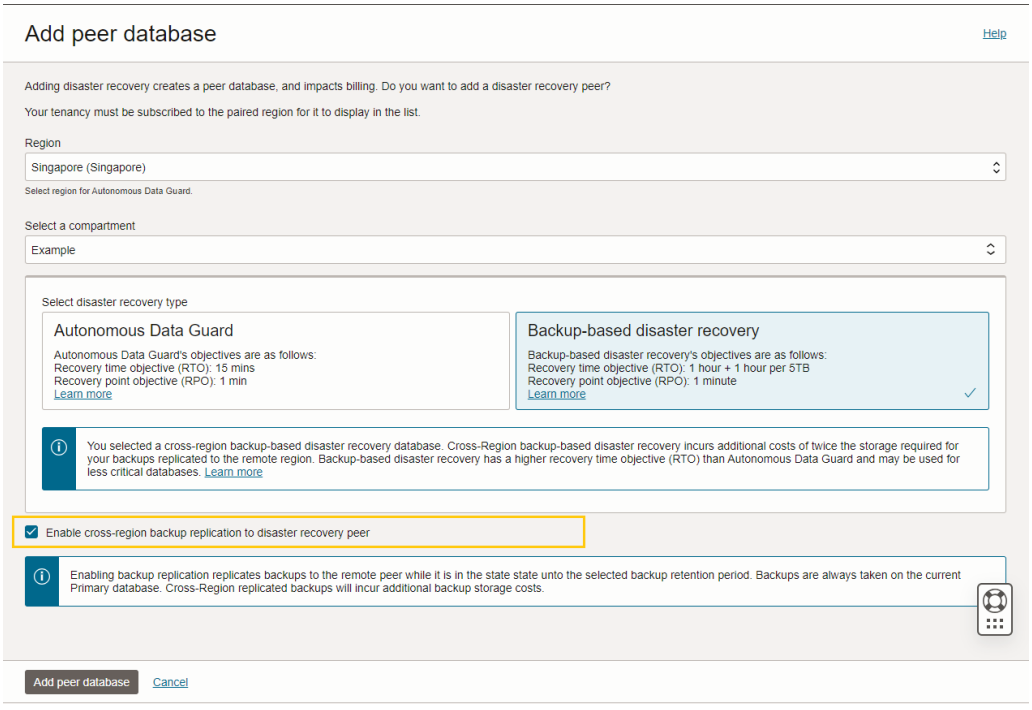
1. On the Autonomous Database Details page, under **Disaster recovery**, in the **Cross-region** field, click **Add peer database** link, alternatively in the **Resources** area click **Disaster recovery**.



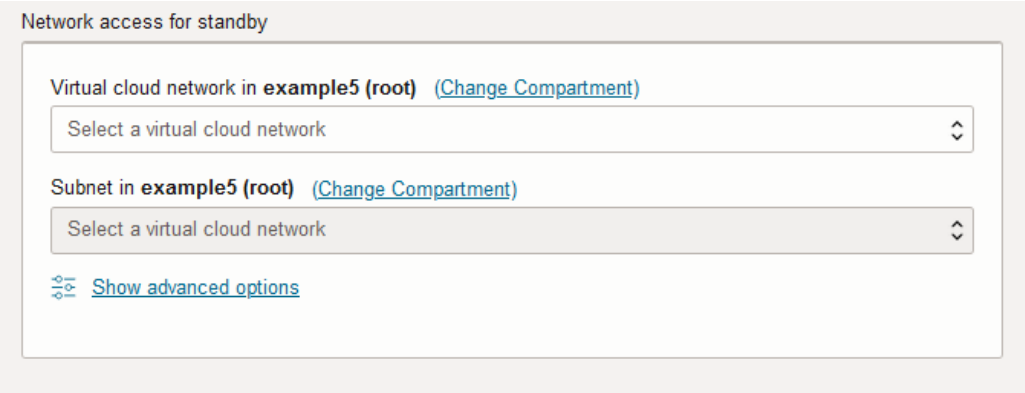


2. Click **Add peer database**.
3. In the **Region** drop-down list, select a region.

   The region list shows the available remote regions where you can create a cross-region peer. When you add a peer database, the list of available regions only shows a remote region if your tenancy is subscribed to the remote region (you must be subscribed to the paired remote region). See Autonomous Database Cross-Region Paired Regions for more information.
4. In the **Select a compartment** drop-down list, select a compartment.
5. Select the disaster recovery type. In addition, when the source database is configured with a private endpoint, enter the private endpoint information for the peer.
   a. Select the disaster recovery type: **Backup-based disaster recovery**.
   b. If you want to enable Cross-Region Backup Replication select the **Enable cross-region backup replication to disaster recovery peer** check-box. See Replicating Backups to a Cross-Region Backup Based Disaster Recovery Peer for more information.



   c. When the source database is configured with a private endpoint, in the **Network access for standby** area enter the **Virtual cloud network** and the **Subnet**.



In these Network access for standby fields you specify the private endpoint's VCN and Subnet on the remote region where the standby is created. Configure Private Endpoints.

> **Note:**
>
> If you change your network access on the source database to enable a private endpoint after the standby is created, you must manually access the standby to enable a private endpoint on the peer.

6. Click **Add peer database**.

   The Autonomous Database Lifecycle State changes to **Updating**. In the **Resources** area, the number next to **Disaster recovery** shows that you now have two peers **(2)**, one local and one remote, and the **State** field shows **Provisioning** for the new peer.

> **Note:**
>
> While you add a new peer, the primary database is available for read/write operations. There is no downtime on the primary database.

Notes for adding a second peer with Backup-Based Disaster Recovery:

- Autonomous Database generates a work request. To view the work request, under **Resources** click **Work Requests**.
- After you add a cross-region (remote) peer, the wallet, and connection string from the primary database will contain only the hostname of the primary database, and the wallet and connection string from the remote database will contain only the hostname of the remote database. This applies to both instance and regional wallets.

  See Cross-Region Disaster Recovery Connection Strings and Wallets for more information.
- If you select **Enable cross-region backup replication to disaster recovery peer**, it can take between several minutes and several hours to replicate the backups to the remote region, depending on the size of the backups. After backups are replicated, when you select **Backups** under **Resources** on the peer database's Oracle Cloud Infrastructure Console, you will see the list of replicated backups.
- When you add a cross region peer there are special considerations when the primary instance is using customer-managed keys, or if you want to switch to use customer-

managed keys. See Autonomous Data Guard with Customer Managed Keys for more information.

- While you add a peer with Backup-Based Disaster Recovery when the **Lifecycle State** field shows **Updating**, the following actions are disabled for the primary database:
  - Move Resource. See Move an Autonomous Database to a Different Compartment for information on moving an instance.
  - Stop. See Stop Autonomous Database for information on stopping an instance.
  - Restart. See Restart Autonomous Database for information on restarting an instance.
  - Restore. See Restore and Recover your Autonomous Database for information on restoring.
- Enable or Disable Backup Replication for an Existing Cross Region Peer

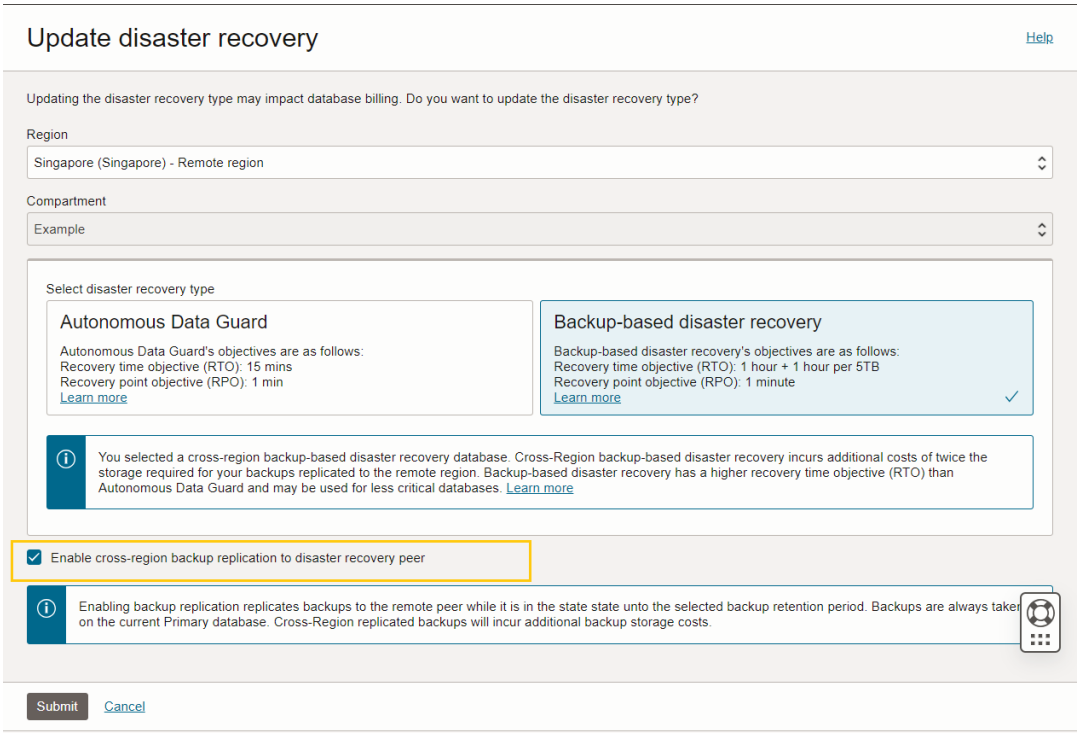
Enable or Disable Backup Replication for an Existing Cross Region Peer

You can enable or disable backup replication on a Backup-Based Disaster Recovery cross-region peer.

To enable or disable backup replication for an existing cross-region Autonomous Data Guard standby:

1. On the **Autonomous Database Details** page, in the **Resources** area select **Disaster recovery**.
2. In a row that lists a cross-region standby, click ⋮ at the end of the row and select **Update disaster recovery**.

   This shows the Update disaster recovery page.



3. Enable or disable backup replication.
   a. If cross-region backup replication is disabled, select **Enable cross-region backup replication to disaster recovery peer** to enable the option.
   b. If cross-region backup replication is enabled, deselect **Enable cross-region backup replication to disaster recovery peer** to disable the option.
4. Click **Submit**.

   The Autonomous Database Lifecycle State changes to **Updating**.

If you select **Enable cross-region backup replication to disaster recovery peer**, it can take between several minutes and several hours to replicate the backups to the remote region, depending on the size of the backups. After backups are replicated, when you select **Backups** under **Resources** on the peer database's Oracle Cloud Infrastructure Console, you will see the list of replicated backups.

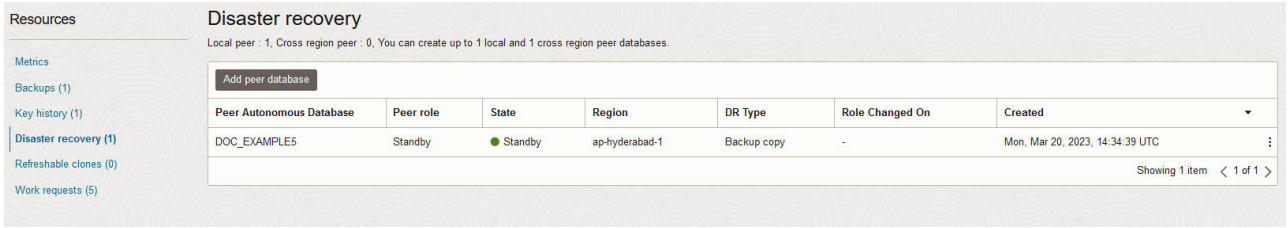
Update Disaster Recovery Type

Describes the steps to change to an alternative Disaster Recovery option.

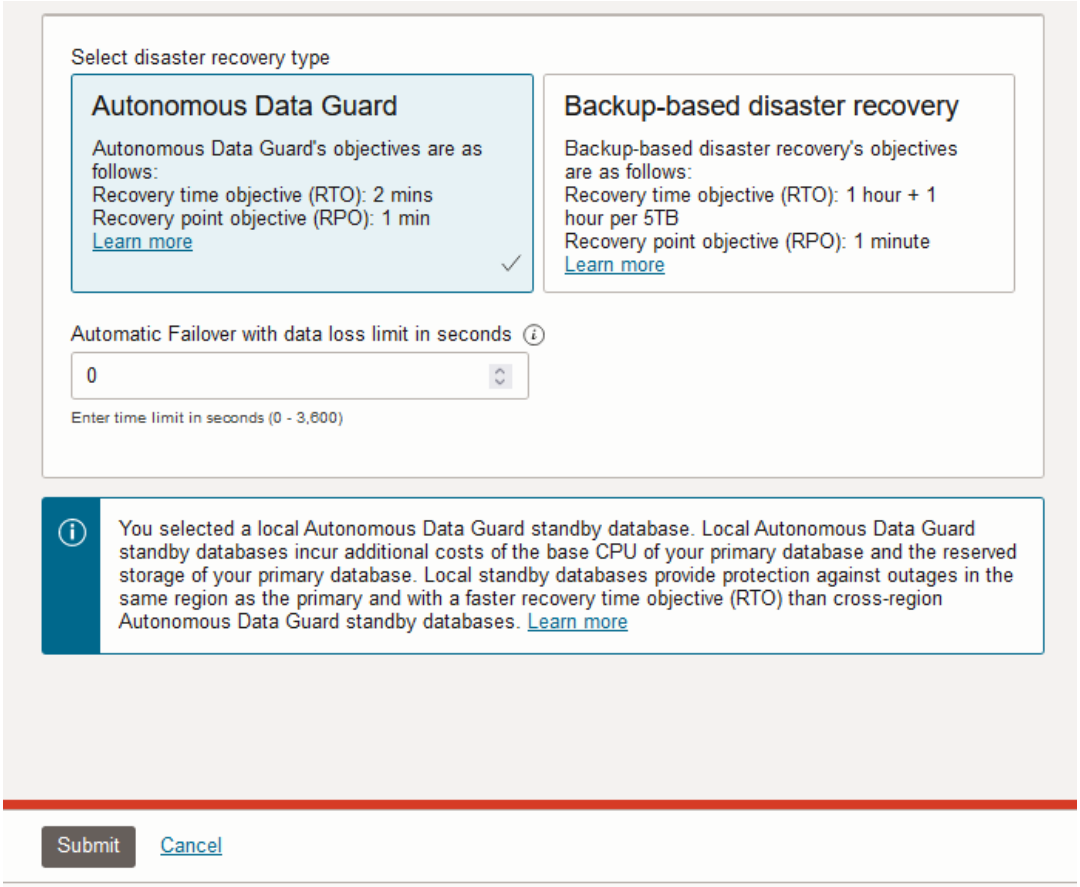
Backup-Based Disaster Recovery is enabled by default for an Autonomous Database with one local peer. Disaster Recovery cannot be disabled for an Autonomous Database instance. However, you can choose to update your Disaster recovery type to Autonomous Data Guard. See Using Standby Databases with Autonomous Database for Disaster Recovery for more information about Autonomous Data Guard.

To update the Disaster recovery type:

1. On the Primary Autonomous Database, on the Autonomous Database Details page under **Resources**, select **Disaster recovery**.



2. In the row showing the database's disaster recovery details, click  at the end of the row and select **Update disaster recovery type**.
3. On the Update disaster recovery type page, choose **Autonomous Data Guard**.



4. Click **Submit**. This starts provisioning an Autonomous Data Guard standby database.
5. The disaster recovery type is changed to, Autonomous Data Guard, as shown in the **DR Type** column.

Disable a Cross-Region (Remote) Peer

Describes the steps to terminate a cross-region (remote) peer.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To terminate a cross-region (remote) peer:

1. On the Primary database, on the Autonomous Database Details page under **Resources**, select Disaster recovery.
2. Access the Oracle Cloud Infrastructure Console for the remote region peer.

   On the Primary database, the Disaster recovery information area shows the **Peer Autonomous Database**. The remote peer has the same display name as the **Primary Database**. However, the display name has an "**_Remote**" extension.

Under **Peer Autonomous Database**, click the link for the remote peer to access the cross-region peer.

3. On the Oracle Cloud Infrastructure Console for the remote peer, on the **Details** page, from the **More actions** drop-down list, select **Terminate**.
4. On the **Terminate Autonomous Database** page, enter the database name to confirm that you want to terminate the cross-region peer.
5. Click **Terminate Autonomous Database**.

   While the peer is terminating, the **Lifecycle State** changes to **Terminating**.

There are limitations for disabling when an instance has a cross-region Backup-Based Disaster Recovery peer, as follows:

- A peer in the remote region cannot be disabled from the primary database.
- When Backup-Based Disaster Recovery is enabled with a cross-region peer, you must terminate the cross-region peer before you terminate the primary role database. If you attempt to terminate the primary, the system gives you an error.

  In this case, after you terminate the cross-region (remote) peer you can terminate the primary database.

Perform a Switchover to a Backup Copy Peer

When you perform a switchover, the primary database becomes the backup copy and the backup copy becomes the primary database, with no data loss.

A switchover is typically done to test failover to the backup copy for audit or certification reasons, or to test your application's failover procedures when you have added a backup copy peer.

For switchover to a backup copy peer, the **Autonomous Database Details** page, shows the **Switchover** link under **Disaster recovery** and the Oracle Cloud Infrastructure Console on the primary database also shows a **Switchover** link in the **Role** field when both the primary database and a backup copy peer are available. You can perform a switchover when the primary database **Lifecycle State** shows **Available** or **Stopped**, and a backup copy is available (the **State** field shows **Standby**).
To see the peer state, under **Resources** click **Disaster recovery** and for the peer listed in the **Peer Autonomous Database** column, check that the **State** shows **Standby**.

Using the Autonomous Database API, you can initiate the switchover operation at any time. See Use the API for more information.

- Perform a Switchover to a Local Backup Copy Peer
  When you perform a switchover the primary database becomes the peer, and the peer becomes the primary database, with no data loss.
- Perform a Switchover to a Cross-Region Backup Copy Peer
- Notes for Performing a Switchover to a Backup Copy Peer
  Provides notes for Backup-Based Disaster Recovery switchover:

Perform a Switchover to a Local Backup Copy Peer

When you perform a switchover the primary database becomes the peer, and the peer becomes the primary database, with no data loss.

A switchover is typically done to test failover to the peer for audit or certification reasons or to test your application's failover procedures with Backup-Based Disaster Recovery.

For a switchover to a backup copy, the **Autonomous Database Details** page, the Oracle Cloud Infrastructure Console on the database with the **Primary** role shows a **Switchover** link in the **Role** field when both the primary database and a peer are available. You can perform a switchover when the primary database **Lifecycle State** shows **Available** or **Stopped** and a peer is available (the **State** field shows **Standby**).

To see the peer status, under **Resources** click Disaster recovery and for the peer listed in the **Peer Autonomous Database** column, check that the **State** field shows **Standby**.

Using the Autonomous Database API, you can initiate the switchover operation at any time. See Use the API for more information.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To perform a switchover, do the following:

1. On the Autonomous Database Details page, under **Disaster recovery**, in the **Role** field, click **Switchover**.

   As an alternative, to initiate a switchover you can select **More actions** and **Switchover** or select **Disaster recovery** under **Resources** and **Switchover**.
2. In the **Confirm switchover to peer** dialog, in the **Select peer** list, choose the peer to switchover.
3. Enter the database name to confirm that you want to switch over.
4. Click **Confirm switchover to peer**.

   When concurrent operations such as scaling are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

   The database **Lifecycle State** changes to **Updating**. To see the state of the peer, under **Resources**, click **Disaster recovery** where the **State** column shows **Role Change in Progress**.

When the switchover completes, Backup-Based Disaster Recovery does the following:

- The Backup-Based Disaster Recovery resource information is updated to reflect the switchover. Select **Disaster recovery** under **Resources** to see the updated information.
- Autonomous Database reports the time in the **Role Changed On** the field.

Perform a Switchover to a Cross-Region Backup Copy Peer

When you perform a switchover, the primary database becomes the peer database and the peer database becomes the primary database, with no data loss.

> **Note:**
>
> For a cross-region switchover you must initiate the switchover from the cross-region peer.

You have several options to access the cross-region peer:

- Select the remote region in Oracle Cloud Infrastructure Console and then access the peer directly.
- Access the primary, and from the primary database you can access the peer from the Autonomous Database Details page by selecting **Disaster recovery**, under **Resources** and clicking the link for the backup copy peer in the **Peer Autonomous Database** column.

To perform a switchover:

1. On the cross-region peer, on the Autonomous Database Details page, under **Disaster recovery**, in the **Role** field, click **Switchover**.

   As an alternative, in the row showing the database's disaster recovery details, click  at the end of the row and select **Switchover**.
2. In the **Confirm switchover to peer** dialog, enter the peer database name to confirm that you want to switch over.
3. In the **Confirm switchover to peer** dialog, click **Confirm switchover to peer**.

   When concurrent operations such as scaling are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

   The database **Lifecycle State** changes to **Role change in progress**. To see the state of the peer database, under **Resources** click **Disaster recovery**. The state **State** shows **Updating**.

When the switchover completes, Autonomous Database does the following:

- The display name shows the Primary indicator.
- The **Disaster recovery** resource information is updated to reflect the switchover. Under **Resources** select **Disaster recovery** to see the updated information.
- Autonomous Database reports the time of the last switchover when you hover over the  in the **Role** field.

See Notes for Performing a Switchover for more information.

Notes for Performing a Switchover to a Backup Copy Peer

Provides notes for Backup-Based Disaster Recovery switchover:

- For a cross-region switchover, you must initiate the switchover from the cross-region peer.

- During the switchover, most of the actions on the Oracle Cloud Infrastructure Console are not available and the Autonomous Database Information page shows the **Lifecycle State** with the value **Updating**.
- The switchover operation keeps the original state of the Primary database. If the Primary database is stopped when you perform a switchover, the Primary database is stopped after the switchover.
- Autonomous Database generates the Switchover Autonomous Database work request. To view the request, under **Resources** click **Work Requests**.
- After a switchover or failover to the peer, the peer becomes the Primary and the graphs on the **Database Dashboard** card in Database Actions and the Oracle Cloud Infrastructure Metrics display information about the Primary database. The graphs and metrics do not contain information about the database that was the Primary before the switchover or failover operation.
- You cannot cancel a cross-region switchover operation after the switchover begins and the **State** shows **Role change in progress**. Your options to cancel the switchover are:
  - Try or retry a switchover or a failover until the operation succeeds.
  - File a service request at Oracle Cloud Support or contact your support representative.

Perform a Failover

When the primary database goes down, with Backup-Based Disaster Recovery you can perform a manual failover to make the local peer the primary database.

Backup-Based Disaster Recovery does not provide an automatic failover option. If you want to provide automatic failover, where the system monitors the primary instance and automatically fails over to a local standby database in certain scenarios, you need to change the disaster recovery type for the local instance to use Autonomous Data Guard.

With both a local Backup-Based Disaster Recovery peer and a cross-region Backup-Based Disaster Recovery peer, Oracle recommends that you attempt a manual failover to the local peer first (not to the cross-region peer).

Depending on how you enable Backup-Based Disaster Recovery, there are different steps to perform a manual failover to a peer:

- **When you configure Backup-Based Disaster Recovery with only a local peer**:

  When you have a local peer and the switchover is not successful, the Oracle Cloud Infrastructure console shows a banner with information about why the switchover was not successful and the Oracle Cloud Infrastructure console shows a **failover** link in the **Role** field that you can click to initiate a failover to the local peer. The failover link only shows when the Primary database is unavailable and a peer is available. That is, the Primary database **Lifecycle State** field shows **Unavailable** and the local peer is available. Using the API, you can initiate manual failover at any time. See Use the API for information on using the API.

  To see the peer status, under **Resources** click **Disaster recovery** and for the peer listed in the Peer Autonomous Database column check that the **State** field shows **Available** or **Stopped**.

- **When you use Backup-Based Disaster Recovery with both a local peer and a cross-region (remote) peer:**

  With Backup-Based Disaster Recovery enabled with both a local peer and a cross-region peer, and the local peer is available, Oracle recommends that you attempt a manual failover to the local peer first (not to the cross-region peer).

If a local peer is unavailable or a manual failover to the local peer fails, you can perform a manual switchover to the cross-region peer. If the switchover to the cross-region peer fails, on the peer the Oracle Cloud Infrastructure console shows a **failover** link in the **Role** field that you can click to initiate a manual failover to the peer.

When you initiate a manual failover it is failed over to the peer based on the Recovery Time Objective (RTO) and Recovery Point Objective (RPO) targets. See Backup-Based Disaster Recovery Recovery Time Objective (RTO) and Recovery Point Objective (RPO) for more information.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To initiate a manual failover to a cross-region backup copy, do the following:

1. On the peer, perform a switchover. See Perform a Switchover to a Local Backup Copy Peer for details.
2. If the switchover attempt in Step 1 fails, on the peer the **Role** field shows a **Failover** link. On the peer, click the **Failover** link.

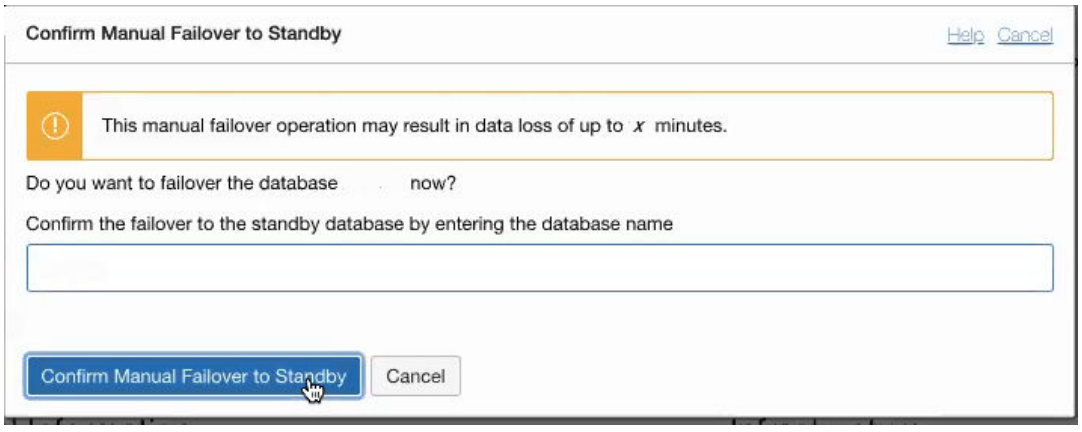
   This shows the **Confirm manual failover to peer** dialog, along with information on possible data loss that may result if you perform the manual failover to the peer.
3. In the **Confirm manual failover to peer** dialog, enter the Autonomous Database name to confirm that you want to failover.
4. In the **Confirm manual failover to peer** dialog, click **Confirm manual failover to peer**.

   When concurrent operations such as scaling are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

To initiate a manual failover when the Primary database is unavailable and the local Peer is available, do the following:

1. On the **Details** page, under **Disaster recovery**, in the **Role** field, click **Failover**.

   This shows the **Confirm manual failover to peer** dialog, along with information on possible data loss that may result if you perform the manual failover to peer.



2. In the **Confirm manual failover to peer** dialog, enter the Autonomous Database name to confirm that you want to failover.
3. In the **Confirm manual failover to peer** dialog, click **Confirm manual failover to peer**.

When the failover completes, Backup-Based Disaster Recovery does the following:

- After a manual failover operation completes you can see any data loss associated with the manual failover in the message on the Oracle Cloud Infrastructure console banner and if you hover over the  in the **Role** field. The manual failover data loss is specified in minutes.
- After a manual failover with Backup-Based Disaster Recovery, if there was a regional failure, when the region comes back online the peer is automatically reconnected or if required reprovisioned.
- After you perform a manual failover to the cross-region peer, the cross-region peer becomes the primary database. In this case, if a local Autonomous Data Guard standby was enabled, a local Standby will be created and attached. If a local Autonomous Data Guard was not enabled before the failover in the primary database, as is the default, you have a local backup copy.

Convert Cross Region Backup-Based Disaster Recovery Peer to Snapshot Standby

A cross-region Backup-Based Disaster Recovery peer can be converted to a snapshot standby. This converts the peer to a read-write database for up to two days.

Snapshot standby CPU usage is billed based on the base CPU count and any additional CPU usage if compute auto scaling is enabled. The number of base CPUs is specified by the number of ECPUs (OCPUs if your database uses OCPUs) as shown in the **ECPU count** or **OCPU count** field on the Oracle Cloud Infrastructure Console.

Snapshot standby storage usage is billed based on the storage of the snapshot standby plus 1 x the storage of the primary database.

Topics

- About Disaster Recovery Snapshot Standby Databases
- Convert Cross Region Disaster Recovery Peer to a Snapshot Standby
- Convert Snapshot Standby Back to a Cross-Region Disaster Recovery Peer

Update Remote Peer Network ACLs for a Backup-Based Disaster Recovery Peer

You can independently modify network ACLs on a remote disaster recovery peer database.

By default the disaster recovery primary and remote peer databases use the same network Access Control Lists (ACLs). Optionally, you can configure ACLs independently on remote peer databases. This provides an option to use different ACLs on remote peer databases.

If you modify the ACLs on a remote peer, Autonomous Database no longer keeps the ACL configuration synchronized between the primary and the remote peer. After you modify the ACLs on a remote peer, Autonomous Database manages the ACLs on the remote peer independently.

To use different network ACLs on a remote Autonomous Database peer:

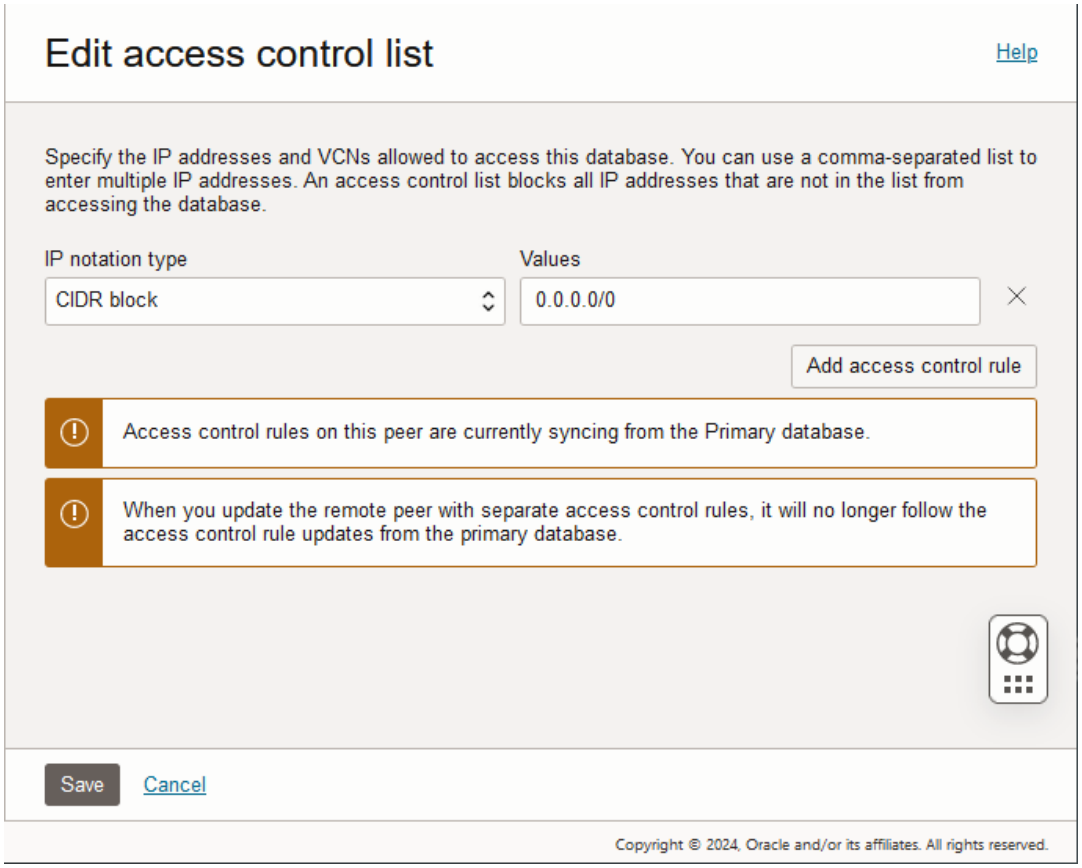
1. On the primary database, on the Autonomous Database Details page, under **Resources** select **Disaster recovery**.
2. Access the remote peer.

The **Disaster recovery** information area shows the **Peer Autonomous Database**. The remote peer database by default has the same display name as the primary database, with an extension. For example, `DBNAME_remote`.

Under **Peer Autonomous Database**, click the link to access a cross-region peer.

3. On the remote peer database, edit the access control list.

   Before you change ACL the dialog shows a message indicating that ACLs on the peer database are syncing from the Primary database. For example:



   See Configure Access Control Lists for an Existing Autonomous Database Instance for more information.

4. Add, remove, or modify one or more ACLs.
5. Click **Save**.

After you modify ACLs, the ACLs on the primary and on the remote peer are managed separately.

If you want to restart the synchronization of ACLs between the primary and the remote peer, you have two options:

- Terminate the peer Autonomous Database and create a new cross region disaster recovery peer database.

  See Disable a Cross-Region (Remote) Peer for details on terminating a remote peer database.

- Contact Oracle Cloud Support and file a service request or contact your support representative.

Use the API

Provides links for details on using API operations to manage Backup-Based Disaster Recovery.

For information about using the API and signing requests, see REST APIs and Security Credentials.

For information about SDKs, see Software Development Kits and Command Line Interface.

Use these API operations to manage Disaster Recovery:

- To enable or disable Backup-Based Disaster Recovery, use UpdateAutonomousDatabase.
- To initiate a manual failover operation, use FailOverAutonomousDatabase.
- To initiate a switchover operation, use SwitchOverAutonomousDatabase.

Use these Terraform APIs to manage Autonomous Database resources:

For information about Terraform, see Terraform Provider and for information about Terraform APIs, see Data Source: oci_database_autonomous_database.

Backup-Based Disaster Recovery Events

You can use Oracle Cloud Infrastructure events to respond when Autonomous Database changes its state due to a Backup-Based Disaster Recovery related event.

Autonomous Database events include the following:

- Begin manual failover
- End manual failover with failover result of success or failure.
- Begin switchover
- End switchover with switchover result of success or failure.

Based on events you can perform actions or send notifications. See Events and Notifications for a Standby Database for more information on using events and producing notifications.

Track Table Changes with Flashback Time Travel

Use Flashback Time Travel to view past states of database objects or to return database objects to a previous state without using point-in-time media recovery.

- About Flashback Time Travel
  Flashback Time Travel lets you track and store transactional changes to a table over its lifetime. Flashback Time Travel is useful for compliance with record stage policies and audit reports. You can also use Flashback Time Travel in various scenarios such as enforcing digital shredding, accessing historical data, and selective data recovery.
- Enable Flashback Time Travel for a Table
  Describes steps to enable Flashback Time Travel for a table in Autonomous Database.
- Disable Flashback Time Travel for a Table
  Describes procedure to disable Flashback Time Travel for a table in Autonomous Database.
- Modify the Retention Time for Flashback Time Travel
  Describes procedure to modify the retention time for Flashback Time Travel in Autonomous Database.

- Purge Historical Data for Flashback Time Travel
  Describes procedure to purge historical data for Flashback Time Travel in Autonomous Database.
- View Flashback Time Travel Information
  Describes data dictionary views to view information about Flashback Time Travel files in Autonomous Database.
- Notes for Flashback Time Travel
  Provides notes and restrictions for using Flashback Time Travel on Autonomous Database.

About Flashback Time Travel

Flashback Time Travel lets you track and store transactional changes to a table over its lifetime. Flashback Time Travel is useful for compliance with record stage policies and audit reports. You can also use Flashback Time Travel in various scenarios such as enforcing digital shredding, accessing historical data, and selective data recovery.

Flashback Time Travel was called Flashback Data Archive in previous Oracle Database versions. The Flashback Time Travel APIs and some Flashback Time Travel terms such as the `FLASHBACK ARCHIVE` privilege retain the Flashback Data Archive naming.

Each Autonomous Database instance has one Flashback Archive named `flashback_archive` that supports Flashback Time Travel operations. The default retention time for `flashback_archive` is 60 days. See Modify the Retention Time for Flashback Time Travel for information on changing the default retention time.

There are restrictions to enable Flashback Time Travel for a table, including the following:

- You must have the `FLASHBACK ARCHIVE` object privilege for `flashback_archive`. By default the `ADMIN` user has this privilege.
- The table must not be any of the following:
  - A nested table
  - A temporary table
  - A remote table
  - An external table
- The table cannot contain `LONG` columns.
- The table cannot contain nested columns.

See Notes for Flashback Time Travel for a list of additional Flashback Time Travel restrictions.

See Using Oracle Flashback Technology for more information on Flashback Technology.

Enable Flashback Time Travel for a Table

Describes steps to enable Flashback Time Travel for a table in Autonomous Database.

By default, Flashback Time Travel is disabled for new and existing tables.

> **Note:**
>
> Flashback Time Travel is not supported on Autonomous Database with database version Oracle Database 21c.

The following is the prerequisite to enable Flashback Time Travel:

- You must be logged in as the `ADMIN` user or have the `FLASHBACK ARCHIVE` object privilege.

  The `ADMIN` user or the user having the `FLASHBACK ARCHIVE` privilege `WITH GRANT OPTION` can grant the `FLASHBACK ARCHIVE` object privilege to another user.

  For example:

  ```
  GRANT FLASHBACK ARCHIVE ON FLASHBACK_ARCHIVE TO <username>


  GRANT FLASHBACK ARCHIVE ON FLASHBACK_ARCHIVE TO <username> WITH GRANT
  OPTION
  ```

  See Manage User Roles and Privileges on Autonomous Database for more information.

You can enable Flashback Time Travel for an existing table or a new table that you are creating.

1. To enable Flashback Time Travel when you create a table:

   ```
   CREATE TABLE employee (EMPNO NUMBER(4) NOT NULL, ENAME VARCHAR2(10), JOB
   VARCHAR2(9), MGR NUMBER(4)) FLASHBACK ARCHIVE;
   ```

2. To enable the Flashback Time Travel for an existing table:

   ```
   ALTER TABLE departments FLASHBACK ARCHIVE;
   ```

After you enable Flashback Time Travel for a table you can disable Flashback Time Travel. See Disable Flashback Time Travel for a Table for more information.

Disable Flashback Time Travel for a Table

Describes procedure to disable Flashback Time Travel for a table in Autonomous Database.

After Flashback Time Travel is enabled for a table you can disable it as the `ADMIN` user or if you have the `FLASHBACK ARCHIVE` object privilege.

For example, to disable Flashback Time Travel for the `employee` table:

```
ALTER TABLE employee NO FLASHBACK ARCHIVE;
```

Modify the Retention Time for Flashback Time Travel

Describes procedure to modify the retention time for Flashback Time Travel in Autonomous Database.

You can modify the retention time for Flashback Time Travel if you are logged in as `ADMIN` user or if you have `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

For example, to set the retention time to 365 days:

```
BEGIN
  DBMS_CLOUD_ADMIN.SET_FLASHBACK_ARCHIVE_RETENTION
   (retention_days => 365);
```

```
END;
/
```

See SET_FLASHBACK_ARCHIVE_RETENTION Procedure for detailed information about the procedure.

Purge Historical Data for Flashback Time Travel

Describes procedure to purge historical data for Flashback Time Travel in Autonomous Database.

You can purge historical data for Flashback Time Travel if you are logged in as the `ADMIN` user or if you have `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

For example, to purge all Flashback Time Travel historical data:

```
BEGIN
  DBMS_CLOUD_ADMIN.PURGE_FLASHBACK_ARCHIVE
   (scope => 'ALL');
END;
/
```

Example to purge the historical Flashback Time Travel data before a specified timestamp:

```
BEGIN
  DBMS_CLOUD_ADMIN.PURGE_FLASHBACK_ARCHIVE
   (scope => 'TIMESTAMP', before_timestamp => '12-JUL-2023 10:24:00');
END;
/
```

Example to purge Flashback Time Travel historical data older than 1 day:

```
BEGIN
  DBMS_CLOUD_ADMIN.PURGE_FLASHBACK_ARCHIVE
   (scope => 'TIMESTAMP', before_timestamp => SYSTIMESTAMP - INTERVAL '1'
DAY);
END;
/
```

Example to purge the historical Flashback Time Travel data before the specified scn:

```
BEGIN
  DBMS_CLOUD_ADMIN.PURGE_FLASHBACK_ARCHIVE
   (scope => 'SCN',before_scn=> '5928826');
END;
/
```

See PURGE_FLASHBACK_ARCHIVE Procedure for detailed information about the procedure.

View Flashback Time Travel Information

Describes data dictionary views to view information about Flashback Time Travel files in Autonomous Database.

You can view information about Flashback Time Travel files in static data dictionary views.

View	Description
`*_FLASHBACK_ARCHIVE`	Displays information about Flashback Time Travel files.
`*_FLASHBACK_ARCHIVE_TS`	Displays tablespaces of Flashback Time Travel files.
`*_FLASHBACK_ARCHIVE_TABLES`	Displays information about tables that have Flashback Time Travel enabled.

Notes for Flashback Time Travel

Provides notes and restrictions for using Flashback Time Travel on Autonomous Database.

Note the following Flashback Time Travel restrictions:

- Flashback Time Travel is not supported on Autonomous Database with database version Oracle Database 21c.
- You cannot enable Flashback Time Travel on tables with `LONG` data type or nested table columns.
- You cannot enable Flashback Time Travel on a nested table, temporary table, external table, materialized view, Advanced Query (AQ) table, hybrid partitioned tables, or non-table object.
- Flashback Time Travel does not support DDL statements that move, split, merge, or coalesce partitions or sub partitions, move tables, or convert `LONG` columns to `LOB` columns.
- Adding or enabling a Constraint, including Foreign Key Constraint, on a table that has been enabled for Flashback Time Travel fails with ORA-55610.

  Dropping or disabling a Constraint (including Foreign Key Constraint) on a table that has been enabled for Flashback Time Travel is supported.
- After enabling Flashback Time Travel on a table, Oracle recommends initially waiting at least 20 seconds before inserting data into the table and waiting up to 5 minutes before using Flashback Query on the table.
- There is one Flashback Data Archive per Autonomous Database instance, named `flashback_archive`, and you cannot create additional Flashback Data Archives.
- You cannot drop the Flashback Data Archive, `flashback_archive`, within an Autonomous Database instance.
- You cannot create, modify, or drop tablespaces for the Flashback Data Archive. Hence, you cannot run these statements:
  - `ALTER FLASHBACK ARCHIVE flashback_archive ADD TABLESPACE;`
  - `ALTER FLASHBACK ARCHIVE flashback_archive MODIFY TABLESPACE;`
  - `ALTER FLASHBACK ARCHIVE flashback_archive REMOVE TABLESPACE;`

- If you enable Flashback Time Travel on a table, but Automatic Undo Management (AUM) is disabled, error ORA-55614 occurs when you try to modify the table.
- To enable Flashback Time Travel on a table, the table cannot use any of the following Flashback Time Travel reserved words as column names:
  - `STARTSCN`
  - `ENDSCN`
  - `RID`
  - `XID`
  - `OP`
  - `OPERATION`

Security

Autonomous Database stores all data with inbuilt security. Only authenticated users and applications can access the data when they connect to the database.

- Security Features Overview
  Describes some of the robust security features that Autonomous Database provides.
- Security and Authentication in Oracle Autonomous Database
  Oracle Autonomous Database stores all data in encrypted format in the database. Only authenticated users and applications can access the data when they connect to the database.
- Security Testing Policies on Autonomous Database
  Autonomous Database is subject to the Oracle Cloud Security Testing policy which describes when and how you may conduct certain types of security testing of Oracle Cloud Infrastructure services, including vulnerability and penetration tests, as well as tests involving data scraping tools.
- Manage Encryption Keys on Autonomous Database
  Describes how to use customer-managed encryption keys with Autonomous Database, and if you are using customer-managed encryption keys, shows how to rotate the keys, switch to Oracle-managed encryption keys, or view the encryption key history.
- Configure Network Access with Access Control Rules (ACLs) and Private Endpoints
  Provides details on how to configure network access with access control rules or using a private endpoint and describes the secure client connection options. Also describes how to enable support for TLS connections (require mutual TLS only or allow both mutual TLS and TLS authentication).
- Audit Autonomous Database
  Autonomous Database provides auditing that allows you to monitor Oracle database activities.
- Rotate Wallets for Autonomous Database
- Use Oracle Database Vault with Autonomous Database
  Oracle Database Vault implements powerful security controls for your database. These unique security controls restrict access to application data by privileged database users, reducing the risk of insider and outside threats and addressing common compliance requirements.
- IAM Policies for Autonomous Database
  Provides information on IAM policies required for API operations on Autonomous Database.

- Use Oracle Data Safe with Autonomous Database
  Provides information on using Oracle Data Safe on Autonomous Database.
- Break Glass Access for SaaS on Autonomous Database
  Autonomous Database supports break glass access for SaaS providers. Break glass access allows a SaaS operations team, when explicitly authorized by a SaaS customer, to access a customer's database to perform critical or emergency operations.
- Encrypt Data While Exporting or Decrypt Data While Importing
  For greater security you can encrypt data that you export to Object Storage. When data on Object Storage is encrypted you can decrypt the data you import or when you use the data in an external table.
- Manage Oracle Cloud Operator Access
  Describes how to grant temporary access to your database for an Oracle Cloud Operator.

Security Features Overview

Describes some of the robust security features that Autonomous Database provides.

Autonomous Database security features include:

- Autonomous Database meets a broad set of international and industry-specific compliance standards, and as part of Oracle Cloud Infrastructure Autonomous Database has achieved attestations for the common compliance frameworks providing an independent assessment of the service's security, privacy, and compliance controls.

  See Regulatory Compliance Certification for more information.
- Autonomous Database applies security patches automatically as soon as they become available.

  See Configuration Management for more information.
- Autonomous Database is subject to the Oracle Cloud Security Testing policy, which describes when and how you may conduct certain types of security testing of Oracle Cloud Infrastructure services, including vulnerability and penetration tests and tests involving data scraping tools.

  See Security Testing Policies on Autonomous Database for more information.
- Autonomous Database provides end-to-end encryption out of the box for the database, backups, and all network communication. All your data, including backups, are encrypted with AES256. You can use Oracle-managed or customer-managed keys to encrypt your data.

  See Manage Encryption Keys on Autonomous Database for more information.
- All network connections are encrypted using TLS 1.2. You can use mutual TLS or one-way TLS connections.
- Autonomous Database provides several options to control client access to your database. You can use public endpoints with access control lists to specify which clients can connect. You can also use private endpoints to place the database in your VCN and use security lists and network security groups to control access to the database.

  See Client Access Control for more information.
- Autonomous Database provides fully automated immutable backups that cannot be tampered with by the users in your tenancy.

  See About Backup and Recovery on Autonomous Database for more information.

- Autonomous Database provides several user authentication methods. You can use local database user names and passwords or external authentication methods, including:
  - Oracle Cloud Infrastructure Identity and Access Management
  - Microsoft Active Directory
  - Azure Active Directory
  - Kerberos

  See Manage Users for more information.
- You can configure Oracle Database Vault to control access to specific schemas from privileged database users such as the ADMIN user.

  See Use Oracle Database Vault with Autonomous Database for more information,
- Autonomous Database provides robust auditing capabilities that enable you to track who did what on the service and on specific databases. You can configure database auditing to audit all actions, such as access to specific objects, schema changes, logons by specific users, and much more.

  See Auditing Overview on Autonomous Database for more information.
- Oracle Cloud Operators do not have authorization to access your data or any other information in your database schemas. When access to your database schemas is required to troubleshoot or mitigate an issue, you can allow a cloud operator to access your Autonomous Database schemas for a limited time.

  See Manage Oracle Cloud Operator Access for more information.

See Security and Authentication in Oracle Autonomous Database for more information.

Security and Authentication in Oracle Autonomous Database

Oracle Autonomous Database stores all data in encrypted format in the database. Only authenticated users and applications can access the data when they connect to the database.

> **Note:**
>
> Oracle Autonomous Database supports the standard security features of the Oracle Database including privilege analysis, network encryption, centrally managed users, secure application roles, transparent sensitive data protection, and others. Additionally, Oracle Autonomous Database adds Label Security, Database Vault, Data Safe, and other advanced security features at no additional cost.

- Configuration Management
  Oracle Autonomous Database provides standard, hardened security configurations that reduce the time and money managing configurations across your databases.
- Data Encryption
  Oracle Autonomous Database uses always-on encryption that protects data at rest and in transit. Data at rest and in motion is encrypted by default. Encryption cannot be turned off.
- Data Access Control
  Securing access to your Oracle Autonomous Database and your data uses several different kinds of access control:

- Auditing Overview on Autonomous Database
  Oracle Autonomous Database provides robust auditing capabilities that enable you to track who did what on the service and on specific databases. Comprehensive log data allows you to audit and monitor actions on your resources, which helps you to meet your audit requirements while reducing security and operational risk.
- Assessing the Security of Your Database and its Data
  Oracle Autonomous Database integrates with Oracle Data Safe to help you assess and secure your databases.
- Regulatory Compliance Certification
  Oracle Autonomous Database meets a broad set of international and industry-specific compliance standards.

Configuration Management

Oracle Autonomous Database provides standard, hardened security configurations that reduce the time and money managing configurations across your databases.

Security patches and updates are done automatically, so you don't spend time, money, or attention to keeping security up to date. These capabilities protect your databases and data from costly and potentially disastrous security vulnerabilities and breaches.

Data Encryption

Oracle Autonomous Database uses always-on encryption that protects data at rest and in transit. Data at rest and in motion is encrypted by default. Encryption cannot be turned off.

Encryption of Data at Rest

Data at rest is encrypted using TDE (Transparent Data Encryption), a cryptographic solution that protects the processing, transmission, and storage of data. Using AES256 tablespace encryption, each database has its own encryption key, and any backups have their own different encryption keys.

By default, Oracle Autonomous Database creates and manages all the master encryption keys used to protect your data, storing them in a secure PKCS 12 keystore on the same systems where the database resides. If your company security policies require, Oracle Autonomous Database can instead use keys you create and manage in the Oracle Cloud Infrastructure Vault service. For more information, see About Master Encryption Key Management on Autonomous Database.

Additionally, customer-managed keys can be rotated when needed in order to meet your organization's security policies.

Note: When you clone a database, the new database gets its own new set of encryption keys.

Encryption of Data in Transit

Clients (applications and tools) connect to Oracle Autonomous Database using supported protocols including SQL*Net, JDBC, and ODBC.

TCPS (Secure TCP) database connection services use the industry-standard TLS 1.2 (Transport Layer Security) protocol for connections and symmetric-key data encryption.

- With mTLS connections, Oracle Autonomous Database users download a connection wallet that contains all necessary files for a client to connect using TCPS. Distribute this wallet only to those users who need and are permitted to have database access. The

client-side configuration uses information in the wallet to perform symmetric-key data encryption.

- Autonomous Database by default supports Mutual TLS (mTLS) connections. You have the option to configure an Autonomous Database instance to allow both mTLS and TLS connections. Using TLS connections, some clients, such as JDBC Thin Driver clients, do not need to download a wallet if you use a TLS connection string and TLS is enabled for the Autonomous Database instance.

See Secure Connections to Autonomous Database for more information.

Data Access Control

Securing access to your Oracle Autonomous Database and your data uses several different kinds of access control:

- Client Access Control
  Client access control for an Autonomous Database instance is enforced by network access control policies, through client connection protocols, and by the access rights of the database user the client uses to connect.
- Database User Access Control
  The Oracle Autonomous Database is configured with an administrative account, ADMIN, that is used to create and manage other database accounts. Oracle Autonomous Database provides a robust set of features and controls including system and object privileges and roles. User profiles allows you to customize password policies to define and implement a secure database user access strategy.
- Oracle Cloud User Access Control
  You use Identity and Access Management (IAM) services to control the privileges of your Oracle Cloud users by specifying the actions those users can perform on your Oracle Autonomous Database.
- Authorized Access on Autonomous Database
  Only authorized users are allowed access to an Autonomous Database instance.
- Autonomous Database Fully Managed Service
  Autonomous Database is a fully managed service and Oracle uses its own Oracle Cloud Infrastructure tenancies to run the Autonomous Database service.

Client Access Control

Client access control for an Autonomous Database instance is enforced by network access control policies, through client connection protocols, and by the access rights of the database user the client uses to connect.

Network Access Control

You define network access control when you set up and configure your Oracle Autonomous Database. There are two options to consider.

- **Private Endpoints and Security Lists:** This is the recommended option. Create your Oracle Autonomous Database in your virtual cloud network (VCN) using private endpoints. You control access to your database using security lists and network security groups allowing you to specify who can create connections to your database.

  For detailed information about creating these resources, see Configure Network Access with Private Endpoints.
- **Public Endpoints and Access Control Lists:** Create your Oracle Autonomous Database using public endpoints allowing access from any client with client credentials. You control

access to your database using network access control lists (ACLs) allowing you to specify IP addresses, CIDR blocks, or VCNs that can connect to your database. Public IPs are easier to discover and attack, and Private Endpoints are recommended where possible.

For detailed information about setting up an ACL, see Configure Access Control Lists for an Existing Autonomous Database Instance.

Client Connection Control

Clients connect through a TCPS (Secure TCP) database connection using standard TLS 1.2 to secure the connection. Oracle Autonomous Database uses self-signed certificates. You can rotate the self-signed certificates from the Oracle Cloud Infrastructure console to meet your organization's security compliance needs. See Rotate Wallets with Immediate Rotation.

The access the client has to the database is restricted by the access rights of the database user the client uses to connect.

Database User Access Control

The Oracle Autonomous Database is configured with an administrative account, ADMIN, that is used to create and manage other database accounts. Oracle Autonomous Database provides a robust set of features and controls including system and object privileges and roles. User profiles allows you to customize password policies to define and implement a secure database user access strategy.

For basic information about standard user management, see User Accounts in *Oracle Database Concepts*. For detailed information and guidance, see Managing Security for Oracle Database Users in *Oracle Database Security Guide*.

If your database user access strategy demands more controls than are provided by standard user management, you can configure your Oracle Autonomous Database to use Database Vault to meet more rigorous requirements.

Using Microsoft Active Directory to Manage Database Users

If you use Microsoft Active Directory as a user repository, you can configure your database to authenticate and authorize Microsoft Active Directory users. This integration enables you to consolidate your user repository while still implementing a rigorous database user access strategy, regardless of whether you use standard user management or Database Vault.

For more information about integrating Microsoft Active Directory with your databases, see Use Microsoft Active Directory with Autonomous Database.

Database Vault

Oracle Database Vault comes preconfigured and ready to use. You can use its powerful security controls to restrict access to application data by privileged database users, reducing the risk threats, and addressing common compliance requirements.

You configure controls to block privileged account access to application data and control sensitive operations inside the database. You configure trusted paths to add additional security controls to authorized data access, database objects, and database commands. Database Vault secures existing database environments transparently, eliminating costly and time consuming application changes.

Before using Database Vault, be sure to review Use Oracle Database Vault with Autonomous Database to gain an understanding of the impact of configuring and enabling Database Vault.

For detailed information on implementing Database Vault features, refer to *Oracle Database Vault Administrator's Guide*.

Oracle Cloud User Access Control

You use Identity and Access Management (IAM) services to control the privileges of your Oracle Cloud users by specifying the actions those users can perform on your Oracle Autonomous Database.

The IAM service provides several kinds of components to help you define and implement a secure cloud user access strategy:

- **Compartment**: A collection of related resources. Compartments are a fundamental component of Oracle Cloud Infrastructure for organizing and isolating your cloud resources.
- **Group**: A collection of users who all need the same type of access to a particular set of resources or compartment.
- **Dynamic Group**: A special type of group that contains resources that match rules that you define. Thus, the membership can change dynamically as matching resources are created or deleted.
- **Policy**: A group of statements that specify who can access which resources, and how. Access is granted at the group and compartment level, which means you write a policy statement that gives a specific group a specific type of access to a specific type of resource within a specific compartment.

Of these, the policy is the primary tool you use to control access because it provides the "Who", "How", "What" and "Where" of a single access constraint. A policy statement has this format:

The format of a policy statement is:

```
Allow
  group <group-name>
  to <control-verb>
  <resource-type>
  in compartment <compartment-name>
```

- **`group <group-name>`** specifies the "Who" by providing the name of an existing IAM group.
- **`to <control-verb>`** specifies the "How" using one of these predefined control verbs:
  - **`inspect`**: the ability to list resources of the given type, without access to any confidential information or user-specified metadata that may be part of that resource.
  - **`read`**: `inspect` plus the ability to get user-specified metadata and the actual resource itself.
  - **`use`**: `read` plus the ability to work with existing resources, but not to create or delete them. Additionally, "work with" means different operations for different resource types.
  - **`manage`**: all permissions for the resource type, including creation and deletion.
- **`<resource-type>`** specifies the "What" using a predefined resource-type. The resource-type values for infrastructure resources are:
  - **`autonomous-databases`**
  - **`autonomous-backups`**

  You may create policy statements that refer to the **`tag-namespaces`** resource-type value if tagging is used in your tenancy.

- **`in compartment <compartment-name>`** specifies the "Where" by providing the name of an existing IAM compartment.

For information about how the IAM service and its components work and how to use them, see Overview of Oracle Cloud Infrastructure Identity and Access Management. For quick answers to common questions about IAM, see the Identity and Access Management FAQ.

Authorized Access on Autonomous Database

Only authorized users are allowed access to an Autonomous Database instance.

Oracle Cloud Operators do not have authorization to access your Autonomous Database. When access to your database is required to troubleshoot or mitigate an issue, you can allow a Cloud Operator to access a database for a limited time.

You allow a Cloud Operator to access the database by running the procedure `DBMS_CLOUD_ADMIN.ENABLE_OPERATOR_ACCESS`. This means if you file a service request with Oracle Cloud Support or by contacting your support representative and Oracle Cloud Operators need to access your database, you must also enable operator access by running `DBMS_CLOUD_ADMIN.ENABLE_OPERATOR_ACCESS`.

Each database access by Oracle Cloud Operators is logged with a request ID and reason.

See Manage Oracle Cloud Operator Access and View Oracle Cloud Infrastructure Operations Actions for more information.

Autonomous Database Fully Managed Service

Autonomous Database is a fully managed service and Oracle uses its own Oracle Cloud Infrastructure tenancies to run the Autonomous Database service.

Oracle Cloud Operators do not have access to customer tenancies and cloud operators cannot access the network.

Auditing Overview on Autonomous Database

Oracle Autonomous Database provides robust auditing capabilities that enable you to track who did what on the service and on specific databases. Comprehensive log data allows you to audit and monitor actions on your resources, which helps you to meet your audit requirements while reducing security and operational risk.

- Auditing Service Level Activities
  All actions Oracle Cloud users perform on the resources that make up your deployment of Oracle Autonomous Database are logged by the Audit service, regardless of the interface used: the Oracle Cloud Infrastructure Console, REST API, Command Line Interface (CLI), Software Development Kits (SDK) and so on.
- Auditing Database Activities
  Oracle Autonomous Database configures the autonomous databases you create to use the unified auditing feature of Oracle Database.

Auditing Service Level Activities

All actions Oracle Cloud users perform on the resources that make up your deployment of Oracle Autonomous Database are logged by the Audit service, regardless of the interface used: the Oracle Cloud Infrastructure Console, REST API, Command Line Interface (CLI), Software Development Kits (SDK) and so on.

You can use the Audit service to perform diagnostics, track resource usage, monitor compliance, and collect security-related events. For more information about the Audit service, see Overview of Audit in *Oracle Cloud Infrastructure Documentation*.

Additionally, when users perform operations on your Oracle Autonomous Database, the database publishes *events* to the Oracle Cloud Events service. The Oracle Cloud Events service allows you to create rules to capture these events and perform actions.

For more information about how the Events service works and how to set up the rules and actions it uses, see Overview of Events. For listings of the Oracle Autonomous Database operations that generate events, see Autonomous Database Event Types.

Auditing Database Activities

Oracle Autonomous Database configures the autonomous databases you create to use the unified auditing feature of Oracle Database.

This feature captures audit records from the following sources and gathers them in a single audit trail in a uniform format:

- Audit records (including `SYS` audit records) from unified audit policies and `AUDIT` settings
- Fine-grained audit records from the `DBMS_FGA` PL/SQL package
- Oracle Database Real Application Security audit records
- Oracle Recovery Manager audit records
- Oracle Database Vault audit records
- Oracle Label Security audit records
- Oracle Data Mining records
- Oracle Data Pump
- Oracle SQL*Loader Direct Load

Audit information is retained for up to 14 days, after which it is automatically purged. To retain audit information for longer, and to easily analyze and report on database activity, use Oracle Data Safe (included with your Oracle Autonomous Database subscription).

See About Auditing Autonomous Database for more information.

Assessing the Security of Your Database and its Data

Oracle Autonomous Database integrates with Oracle Data Safe to help you assess and secure your databases.

Oracle Data Safe helps you understand the sensitivity of your data, evaluate risks to data, mask sensitive data, implement and monitor security controls, assess user security, monitor user activity, and address data security compliance requirements in your databases.

You use Oracle Data Safe to identify and protect sensitive and regulated data your Oracle Autonomous Database by registering your database with Data Safe. Then, you use the Data Safe console directly from the Details page of your database.

For more information about using Data Safe, see Use Oracle Data Safe Features.

Regulatory Compliance Certification

Oracle Autonomous Database meets a broad set of international and industry-specific compliance standards.

Certification	Description
C5	The Cloud Computing Compliance Controls Catalog (C5)
CSA STAR	The Cloud Security Alliance (CSA) Security Trust, Assurance and Risk (STAR)
Cyber Essentials (UK) Cyber Essentials Plus (UK)	Oracle Cloud Infrastructure has achieved Cyber Essentials and Cyber Essentials Plus certification in these regions: • UK Gov South (London) • UK Gov West (Newport)
DESC (UAE)	Dubai Electronic Security Center CSP Security Standard
DoD IL4/IL5	DISA Impact Level 5 authorization in these regions: • US DoD East (Ashburn) • US DoD North (Chicago) • US DoD West (Phoenix)
ENS High (Spain)	The Esquema Nacional de Seguridad with the level of accreditation High.
FedRAMP High	Federal Risk and Authorization Management Program (U.S. Government Regions only)
FSI (S. Korea)	The Financial Security Institute
HDS	The French Public Health Code requires healthcare organizations that control, process, or store health or medical data to use infrastructure, hosting, and platform service providers that are Hébergeur de Données de Santé (HDS) accredited and certified
HIPAA	Health Insurance Portability and Accountability Act
HITRUST	The Health Information Trust Alliance
IRAP (Australia)	The Infosec Registered Assessors Program. Sydney and Melbourne regions
ISMS (S. Korea)	The Information Security Management System
ISO/IEC 27001:2013	International Organization for Standardization 27001
ISO/IEC 27017:2015	Code of Practice for Information Security Controls Based on ISO/IEC 27002 for Cloud Services
ISO/IEC 27018:2014	Code of Practice for Protection of Personally Identifiable Information (PII) In Public Clouds Acting as PII Processors
MeitY (India)	The Ministry of Electronics and Information Technology
MTCS (Singapore)	Multi-Tier Cloud Service (MTCS) Level 3
PASF (UK OC4)	Police Assured Secure Facilities (PASF) in these regions: • UK Gov South (London) • UK Gov West (Newport)
PCI DSS	Payment Card Industry Data Security Standard is a set of requirements intended to ensure that all companies that process, store, or transmit credit card information maintain a secure environment
SOC 1	System and Organization Controls 1
SOC 2	System and Organization Controls 2

For more information and a complete list of certifications, see Oracle Cloud Compliance.

Security Testing Policies on Autonomous Database

Autonomous Database is subject to the Oracle Cloud Security Testing policy which describes when and how you may conduct certain types of security testing of Oracle Cloud Infrastructure services, including vulnerability and penetration tests, as well as tests involving data scraping tools.

Oracle regularly performs penetration and vulnerability testing and security assessments against Oracle Cloud Infrastructure, platforms, and applications. These tests are intended to validate and improve the overall security of Oracle Cloud services. You can also conduct your own testing as per the rules explained in the policy.

See Oracle Cloud Security Testing Policies for more information.

Manage Encryption Keys on Autonomous Database

Describes how to use customer-managed encryption keys with Autonomous Database, and if you are using customer-managed encryption keys, shows how to rotate the keys, switch to Oracle-managed encryption keys, or view the encryption key history.

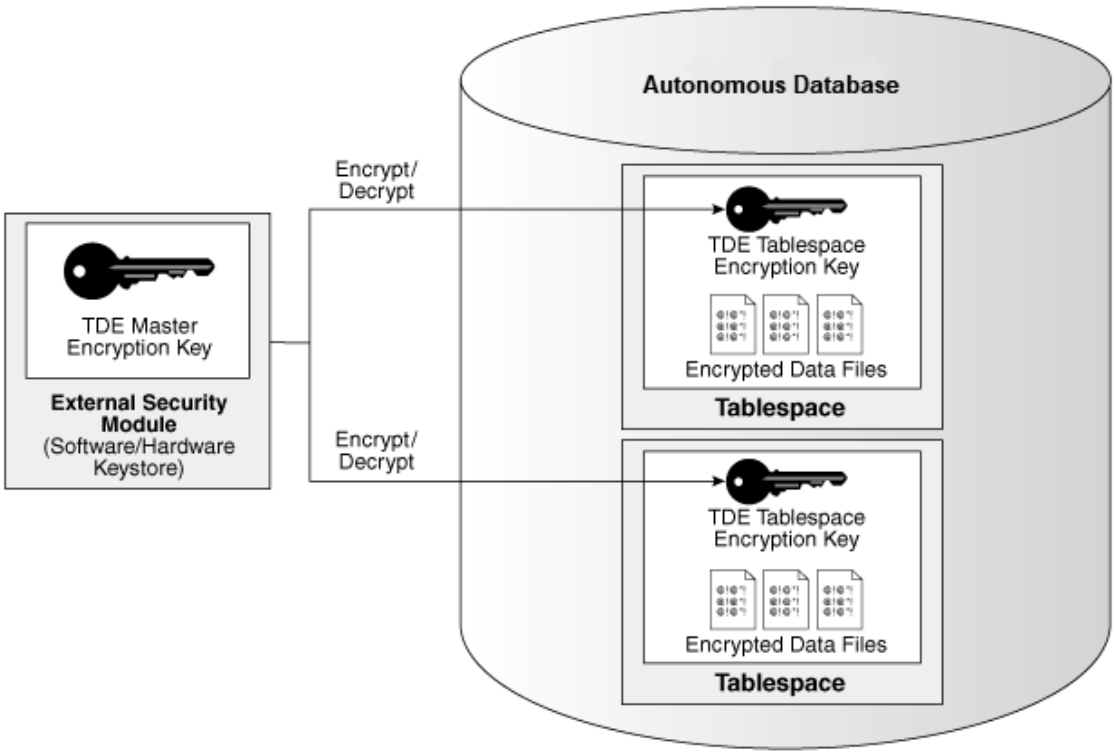
- About Master Encryption Key Management on Autonomous Database
  Autonomous Database provides two options for Transparent Data Encryption (TDE) to encrypt your database: Oracle-managed encryption keys and Customer-managed encryption keys.
- Prerequisites to Use Customer-Managed Encryption Keys on Autonomous Database
  Perform these prerequisite steps to use customer-managed keys on Autonomous Database:
- Use Customer-Managed Encryption Keys with Vault Located in Local Tenancy
  Shows the steps to select customer-managed master encryption keys on Autonomous Database. If you are using customer-managed master encryption keys, follow these steps to rotate the master keys.
- Use Customer-Managed Encryption Key Located in a Remote Tenancy
  Shows the steps to select customer-managed master encryption keys from a Vault on a remote tenancy.
- Switch to Oracle-Managed Encryption Keys on Autonomous Database
  Shows the steps to switch to Oracle-managed master encryption keys on Autonomous Database if you are using customer-managed encryption keys.
- View History for Customer-Managed Encryption Keys on Autonomous Database
  You can view the key history from the Oracle Cloud Infrastructure Console or by selecting from the `V$ENCRYPTION_KEYS` view.
- Notes for Using Customer-Managed Keys with Autonomous Database
  Provides additional information and notes for using customer-managed keys with Autonomous Database

About Master Encryption Key Management on Autonomous Database

Autonomous Database provides two options for Transparent Data Encryption (TDE) to encrypt your database: Oracle-managed encryption keys and Customer-managed encryption keys.

Autonomous Database uses Transparent Data Encryption, including a TDE master key and TDE tablespace keys to encrypt data in the database. As shown in the following figure, the

TDE master key generates and encrypts/decrypts the TDE tablespace keys, and the TDE tablespace keys encrypt the data files.



- Oracle-Managed Master Encryption Keys on Autonomous Database
  By default Autonomous Database uses Oracle-managed encryption keys.
- Customer-Managed Encryption Keys on Autonomous Database
  If your organization's security policies require customer-managed encryption keys, you can configure Autonomous Database to use an Oracle Cloud Infrastructure Vault master encryption key. With customer-managed master encryption keys, Autonomous Database uses the master encryption key to generate the TDE master key.
- About Customer-Managed Encryption Key Rotation on Autonomous Database
  Describes how to rotate customer-managed encryption keys on Autonomous Database.

Oracle-Managed Master Encryption Keys on Autonomous Database

By default Autonomous Database uses Oracle-managed encryption keys.

Using Oracle-managed keys, Autonomous Database creates and manages the encryption keys that protect your data and Oracle handles rotation of the TDE master key.

Customer-Managed Encryption Keys on Autonomous Database

If your organization's security policies require customer-managed encryption keys, you can configure Autonomous Database to use an Oracle Cloud Infrastructure Vault master encryption key. With customer-managed master encryption keys, Autonomous Database uses the master encryption key to generate the TDE master key.

> **Caution:**
>
> The customer-managed encryption key is stored in Oracle Cloud Infrastructure Vault, external to the database host. If the customer-managed encryption key is disabled or deleted, the database will be inaccessible.

Use customer-managed encryption keys by performing the following steps:

1. Create a master encryption key in your Oracle Cloud Infrastructure Vault.

   See Prerequisites to Use Customer-Managed Encryption Keys on Autonomous Database for more information.

2. Select customer-managed encryption keys from the Oracle Cloud Infrastructure Console:
   - For an existing database, select **Manage Encryption Key** on the Oracle Cloud Infrastructure Console.
   - While provisioning, under **Advanced Options**, on the **Encryption Key** tab select either **Encrypt using customer-managed key in this tenancy** or **Encrypt using a customer-managed key located in a remote tenancy**.
   - While cloning, under **Advanced Options**, on the **Encryption Key** tab select either **Encrypt using customer-managed key in this tenancy** or **Encrypt using a customer-managed key located in a remote tenancy**.

About Customer-Managed Encryption Key Rotation on Autonomous Database

Describes how to rotate customer-managed encryption keys on Autonomous Database.

When you rotate the customer-managed master encryption key, Autonomous Database generates a new TDE master key and uses the new TDE master key to re-encrypt the tablespace encryption keys that encrypt and decrypt your data. This operation is fast and does not require database downtime. It does not change the tablespace keys and does not re-encrypt customer data.

> **Note:**
>
> Using the Oracle Cloud Infrastructure Console you can rotate an Oracle Cloud Infrastructure Vault master encryption key with the **Rotate Key** command. This is a separate action and does not result in a new master encryption key for your Autonomous Database. To rotate the master encryption key of your Autonomous Database, create a new master encryption key in Oracle Cloud Infrastructure Vault and follow the steps described below.

To rotate customer-managed encryption keys:

1. Create a new master encryption key in your Oracle Cloud Infrastructure Vault. If you already have multiple master encryption keys, then select a master encryption key that is different than the key you are using as your master encryption key for your Autonomous Database instance.

   See Prerequisites to Use Customer-Managed Encryption Keys on Autonomous Database for more information.

2. Rotate the master encryption key from the Oracle Cloud Infrastructure Console:

See Use Customer-Managed Encryption Keys with Vault Located in Local Tenancy for more information.

Prerequisites to Use Customer-Managed Encryption Keys on Autonomous Database

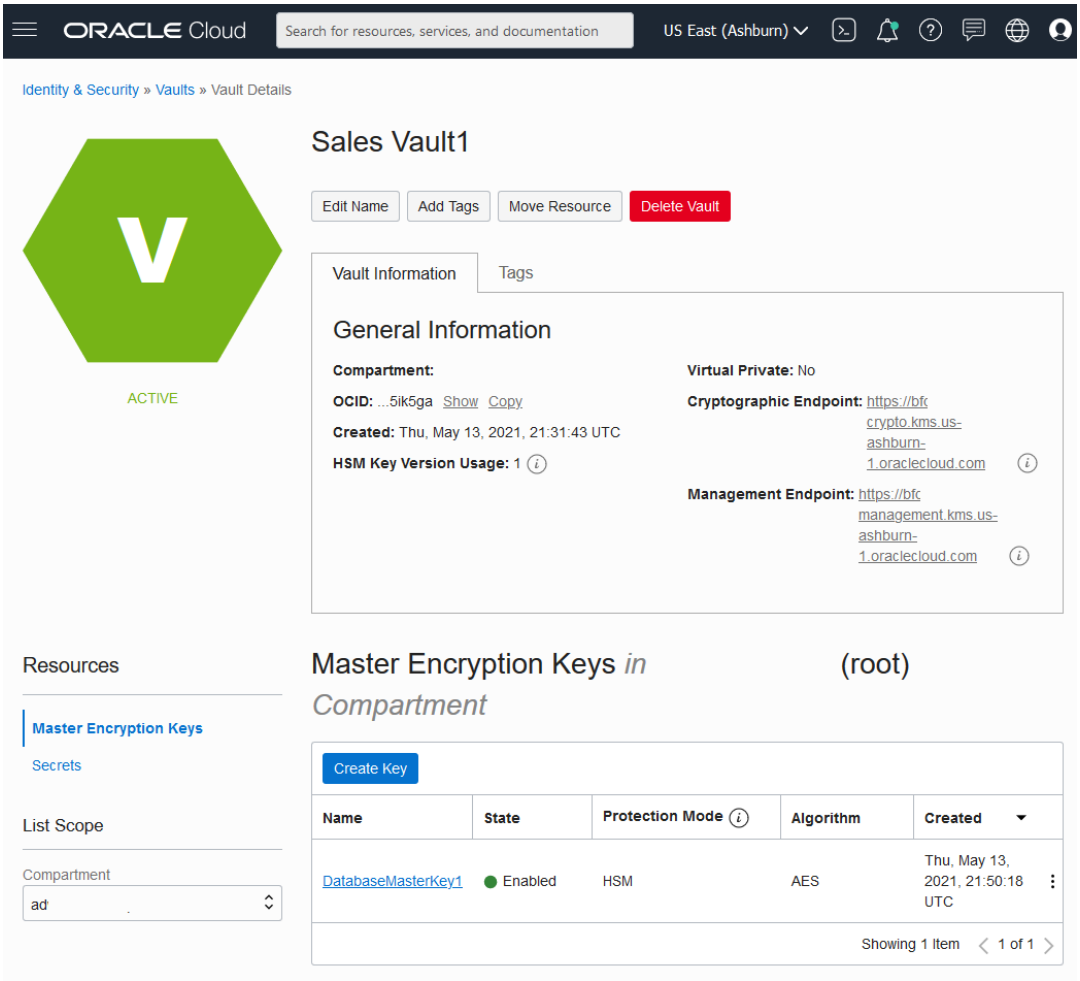
Perform these prerequisite steps to use customer-managed keys on Autonomous Database:

1. Create an Oracle Cloud Infrastructure Vault.
   a. Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
   b. From the Oracle Cloud Infrastructure left navigation menu click **Identity and Security**.
   c. Under **Key Management & Secret Management** click **Vault**.
   d. Select an existing Vault or create a new Vault.

      For more details, see the instructions for creating a vault, To create a new vault.
2. Create a Master Encryption Key in the Vault.

> **Note:**
>
> You must use these options when you create the key:
>
> - **Key Shape: Algorithm**: AES (Symmetric key used for Encrypt and Decrypt)
> - **Key Shape: Length**: 256 bits

For more information, see To create a new master encryption key and Overview of Key Management.



3. Create dynamic group and policy statements for the dynamic group to enable access to Oracle Cloud Infrastructure resources (Vaults and Keys).

   This step depends on whether the vault is on the same tenancy as the Autonomous Database instance or on a different tenancy:

   - Vault and keys are in the same tenancy as the Autonomous Database instance. See Create Dynamic Group and Policies for Customer Managed Keys with Vault in Same Tenancy as Database for more information.
   - Vault and keys are in a different tenancy. See Create Dynamic Group and Policies for Customer Managed Keys with Vault in Different Tenancy than the Database for more information.

You must replicate the vault and keys to use customer-managed encryption keys with Autonomous Data Guard with a remote Standby database. Replication of vaults and keys across regions is only possible if you select the **virtual private vault** option when you create the vault.

See Autonomous Data Guard with Customer Managed Keys for more information.

- Create Dynamic Group and Policies for Customer Managed Keys with Vault in Same Tenancy as Database
  Create dynamic group and policies to provide access to the vault and keys for customer-managed keys when the vault and keys are in the same tenancy as the Autonomous Database instance.

- Create Dynamic Group and Policies for Customer Managed Keys with Vault in Different Tenancy than the Database
  Perform these steps to use customer-managed keys when the Autonomous Database instance and vaults and keys are in different tenancies.

Create Dynamic Group and Policies for Customer Managed Keys with Vault in Same Tenancy as Database

Create dynamic group and policies to provide access to the vault and keys for customer-managed keys when the vault and keys are in the same tenancy as the Autonomous Database instance.

1. Create a dynamic group to make the master encryption key accessible to the Autonomous Database instance.
   a. In the Oracle Cloud Infrastructure console click **Identity & Security**.
   b. Under **Identity** click **Domains** and select an identity domain (or create a new identity domain).
   c. Under **Identity domain**, click **Dynamic groups**.
   d. Click **Create dynamic group** and enter a **Name**, a **Description**, and a rule.
      - **Create Dynamic Group for an existing database:**

        You can specify that an Autonomous Database instance is part of the dynamic group. The dynamic group in the following example includes only the Autonomous Database whose OCID is specified in the `resource.id` parameter:

        ```
        resource.id = '<your_Autonomous_Database_instance_OCID>'
        ```

      - **Create a Dynamic Group for a database that has not been provisioned yet:**

        When you are creating the dynamic group before you provision or clone an Autonomous Database instance, the OCID for the new database is not yet available. For this case, create a dynamic group that specifies the resources in a given compartment:

        ```
        resource.compartment.id = '<your_Compartment_OCID>'
        ```

   e. Click **Create**.
2. Write policy statements for the dynamic group to enable access to Oracle Cloud Infrastructure resources (vaults and keys).
   a. In the Oracle Cloud Infrastructure console click **Identity & Security** and click **Policies**.
   b. To write policies for a dynamic group, click **Create Policy**, and enter a **Name** and a **Description**.
   c. Use the **Policy Builder** to create a policy for vault and keys in the local tenancy.

      For example, the following allows the members of the dynamic group `DGKeyCustomer1` to access the vaults and keys in the compartment named `training`:

      ```
      Allow dynamic-group DGKeyCustomer1 to use vaults in compartment training
      Allow dynamic-group DGKeyCustomer1 to use keys in compartment training
      ```

      This sample policy applies for a single compartment. You can specify that a policy applies for your tenancy, a compartment, a resource, or a group of resources.

To use customer-managed keys with Autonomous Data Guard with a remote standby, the following policy is also required:

```
Allow dynamic-group DGKeyCustomer1 to manage vaults in compartment
training
Allow dynamic-group DGKeyCustomer1 to manage keys in compartment
training
```

d. Click **Create** to save the policy.

See the following for more information:

- Managing Policies
- Autonomous Data Guard with Customer Managed Keys
- Replicating Vaults and Keys

Create Dynamic Group and Policies for Customer Managed Keys with Vault in Different Tenancy than the Database

Perform these steps to use customer-managed keys when the Autonomous Database instance and vaults and keys are in different tenancies.

In this case, you need to supply OCID values when you change to customer-managed keys. In addition, you need to define dynamic groups and policies that allow the Autonomous Database instance to use vaults and keys in a different tenancy.

1. Copy the master encryption key OCID.
2. Copy the vault OCID.
3. Copy the tenancy OCID (the remote tenancy that contains vaults and keys).
4. On the tenancy with the Autonomous Database instance, create a dynamic group.
   a. In the Oracle Cloud Infrastructure console, on the tenancy with the Autonomous Database instance, click **Identity & Security**.
   b. Under **Identity** click **Domains** and select an identity domain (or create a new identity domain).
   c. Under **Identity domain**, click **Dynamic groups**.
   d. Click **Create dynamic group** and enter a **Name**, a **Description**, and a rule.
      - **Create Dynamic Group for an existing database:**

        You can specify that an Autonomous Database instance is part of the dynamic group. The dynamic group in the following example includes only the Autonomous Database whose OCID is specified in the `resource.id` parameter:

        ```
        resource.id = '<your_Autonomous_Database_instance_OCID>'
        ```

      - **Create a Dynamic Group for a database that has not been provisioned yet:**

        When you are creating the dynamic group before you provision or clone an Autonomous Database instance, the OCID for the new database is not yet available. For this case, create a dynamic group that specifies the resources in a given compartment:

        ```
        resource.compartment.id = '<your_Compartment_OCID>'
        ```

e. Click **Create**.

5. On the tenancy with the Autonomous Database instance, define the policies to allow access to vaults and keys (where the vaults and keys are on a different tenancy).

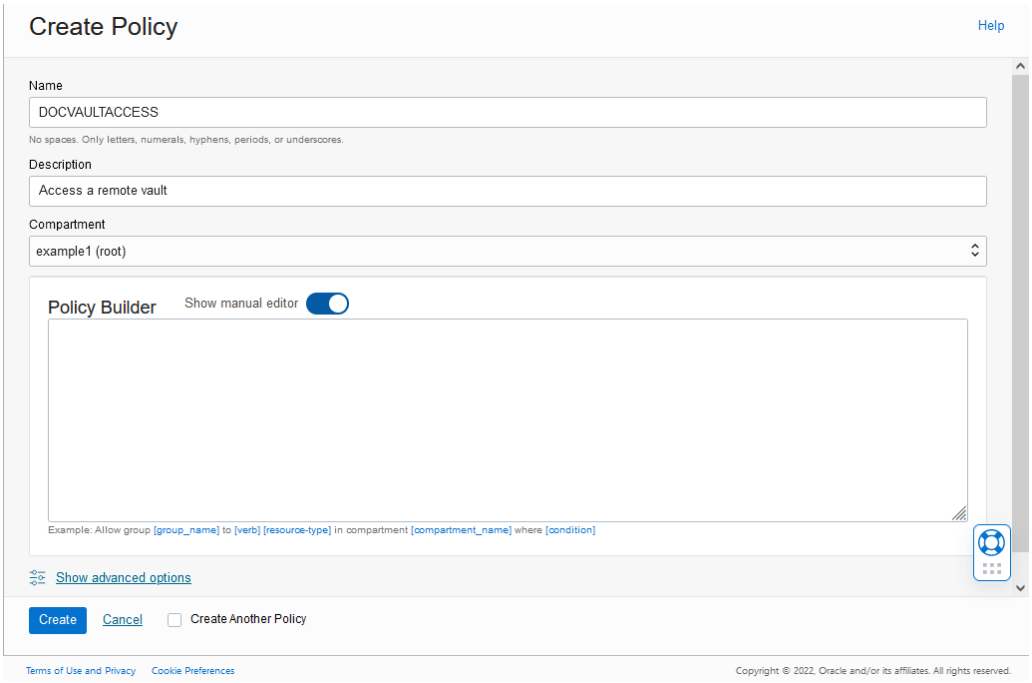
   a. In the Oracle Cloud Infrastructure console click **Identity & Security**.

   b. Under **Identity** click **Policies**.

   c. To write a policy, click **Create Policy**.

   d. On the Create Policy page, enter a Name and a Description.

   e. On the Create Policy page, select **Show manual editor**.



   f. In the policy builder, add policies so that the Autonomous Database instance is able to access vaults and keys located in the different tenancy. Also add policies for the IAM group that the IAM user belongs to so that the Oracle Cloud Infrastructure Console for the Autonomous Database instance can show details about the key that resides in a different tenancy.

   For example, in the generic policy, call the tenancy with the Autonomous Database instance Tenancy-1 and the tenancy with vaults and keys, Tenancy-2:

   Copy the following policy and replace the variables and names with the values you define, where the dynamic group name `ADB-DynamicGroup` is the dynamic group you created in Step 4:

```
define tenancy REMTEN as <ocid of tenancy-2>
endorse dynamic-group ADB-DynamicGroup to use vaults in tenancy REMTEN
endorse dynamic-group ADB-DynamicGroup to use keys in tenancy REMTEN
endorse group MyUserGroup to use vaults in tenancy REMTEN
endorse group MyUserGroup to use keys in tenancy REMTEN
```

For example, the following allows the members of the dynamic group `DGKeyCustomer1` to access the remote vaults and keys in the tenancy named `training2`:

```
define tenancy training2 as ocid1.tenancy.oc1..aaa_example_rcyx2a
endorse dynamic-group DGKeyCustomer1 to use vaults in tenancy training2
endorse dynamic-group DGKeyCustomer1 to use keys in tenancy training2
endorse group MyUserGroup to use vaults in tenancy training2
endorse group MyUserGroup to use keys in tenancy training2
```

g. Click **Create** to save the policy.

6. Copy the tenancy OCID (the tenancy that contains the Autonomous Database instance).
7. Copy the Dynamic Group OCID (for the Dynamic Group you created in Step 4).
8. On the remote tenancy with vaults and keys, define a dynamic group and policies to allow the Autonomous Database instance to access vaults and keys.
   a. In the Oracle Cloud Infrastructure console, click **Identity & Security**.
   b. Under **Identity** click **Policies**.
   c. To create a policy, click **Create Policy**.
   d. On the Create Policy page, enter a Name and a Description.
   e. On the Create Policy page, select **Show manual editor**.
   f. In the policy builder, add policies and a dynamic group to provide access to the dynamic group on the tenancy with the Autonomous Database instance (Tenancy-1), such that the Autonomous Database instance can use the vaults and keys in Tenancy-2. Also need to add policies to allow the user group to access the vault and keys to display information on the Oracle Cloud Infrastructure Console for the Autonomous Database instance in a different tenancy.

      Use the **Policy Builder** to create a dynamic group and a policy for vaults and keys.

      ```
      define tenancy ADBTEN as <ocid of tenancy-1>
      define dynamic-group REM-ADB-DG as <ocid of the Dynamic Group in
      tenancy-1>
      define group REMGROUP as <group-ocid>
      admit dynamic-group REM-ADB-DG of tenancy ADBTEN to use vaults in
      tenancy
      admit dynamic-group REM-ADB-DG of tenancy ADBTEN to use keys in tenancy
      admit group REMGROUP of tenancy ADBTEN to use vaults in tenancy
      admit group REMGROUP of tenancy ADBTEN to use keys in tenancy
      ```

      For example define the following on the remote tenancy to allow the members of the dynamic group `DGKeyCustomer1` and the group `REMGROUP` to access the remote vaults and keys in the tenancy named `training2`:

      ```
      define tenancy adbdemo5 as ocid1.tenancy.oc1..aaa_example_4cnl5q
      define dynamic-group REM-ADB-DG as
      ocid1.dynamicgroup.oc1..aaa_example_526bia
      define group REMGROUP as ocid1.group.oc1..aaa_example_6vctn6xsaq
      admit dynamic-group REM-ADB-DG of tenancy adbdemo5 to use vaults in
      tenancy
      admit dynamic-group REM-ADB-DG of tenancy adbdemo5 to use keys in
      tenancy
      ```

```
admit group REMGROUP of tenancy ADBTEN to use vaults in tenancy
admit group REMGROUP of tenancy ADBTEN to use keys in tenancy
```

g. Click **Create** to save the policy.

Use Customer-Managed Encryption Keys with Vault Located in Local Tenancy

Shows the steps to select customer-managed master encryption keys on Autonomous Database. If you are using customer-managed master encryption keys, follow these steps to rotate the master keys.

> **Caution:**
>
> The customer-managed encryption key is stored in Oracle Cloud Infrastructure Vault, external to the database host. If the customer-managed encryption key is disabled or deleted, the database will be inaccessible.

For details on using customer-managed keys where the Vault is located in a remote tenancy, see Use Customer-Managed Encryption Key Located in a Remote Tenancy.

On Autonomous Database you can choose customer-managed keys as follows:

- While provisioning, under **Advanced Options**, on the **Encryption Key** tab.
- While cloning, under **Advanced Options**, on the **Encryption Key** tab

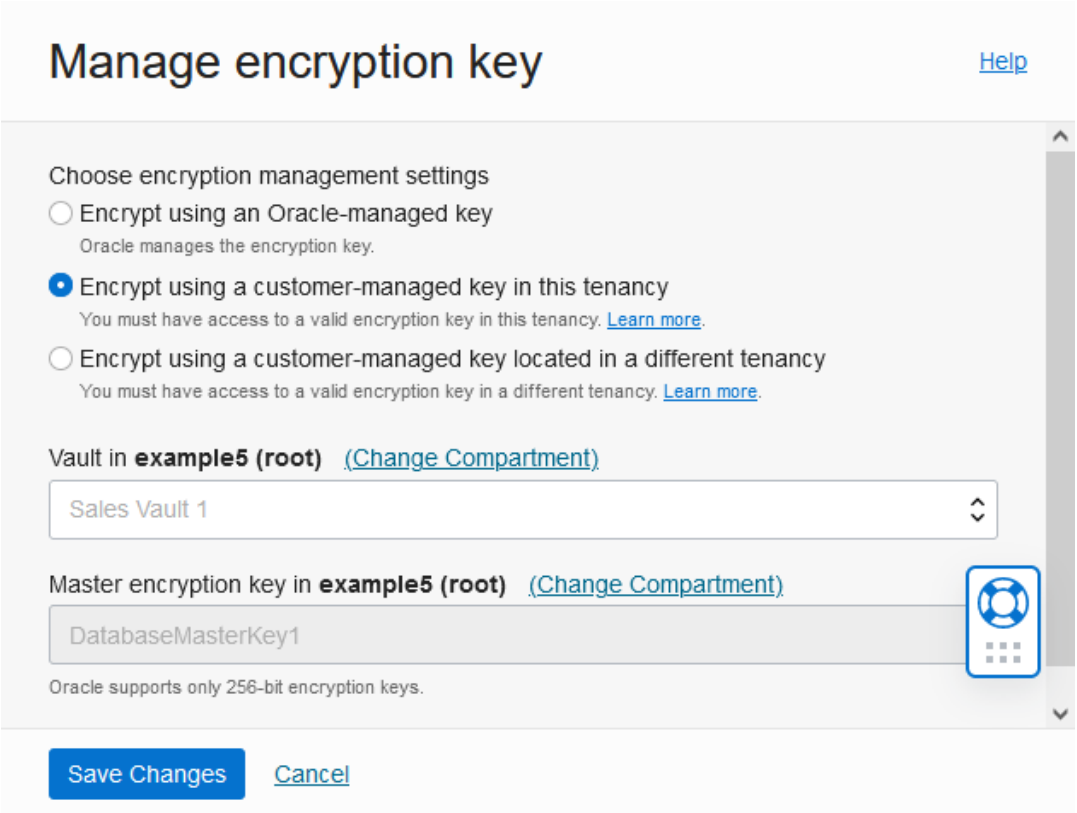
Follow these steps if your Autonomous Database is using Oracle-managed keys and you want to switch to customer-managed encryption keys with the vault in the local tenancy, or if you are using customer-managed encryption keys and you want to rotate the master key.

1. Perform the required customer-managed encryption key prerequisite steps as necessary. See Prerequisites to Use Customer-Managed Encryption Keys on Autonomous Database for more information.
2. On the **Details** page, from the **More actions** drop-down list, select **Manage encryption key**.
3. On the **Manage encryption key** page, select **Encrypt using customer-managed key in this tenancy**.

   If you are already using customer-managed keys and you want to rotate the TDE keys, follow these steps and select a different key (select a key that is different from the currently selected master encryption key).
4. Select a Vault.

   Click **Change Compartment** to select a vault in a different compartment.
5. Select a Master encryption key.

   Click **Change Compartment** to select a master encryption key in a different compartment.



When cross-region Autonomous Data Guard is enabled the **Vault** and **Master encryption key** values show the keys that are replicated in both the primary and the remote standby region.

6. Click **Save Changes**.

The **Lifecycle State** changes to **Updating**. When the request completes, the **Lifecycle State** shows **Available**.

After the request completes, on the Oracle Cloud Infrastructure Console, the key information shows on the Autonomous Database Information page under the heading **Encryption**. This area shows the **Encryption Key** field with a link to the Master Encryption Key and the **Encryption Key OCID** field with the Master Encryption Key OCID.

See Notes for Using Customer-Managed Keys with Autonomous Database for more information.

Use Customer-Managed Encryption Key Located in a Remote Tenancy

Shows the steps to select customer-managed master encryption keys from a Vault on a remote tenancy.

When you use customer-managed master encryption keys with a Vault in a remote tenancy, the Vault and the Autonomous Database instance must be in the same region. To change the tenancy, on the sign-on page click **Change tenancy**. After you change the tenancy, make sure to select the same region for both the Vault and the Autonomous Database instance.

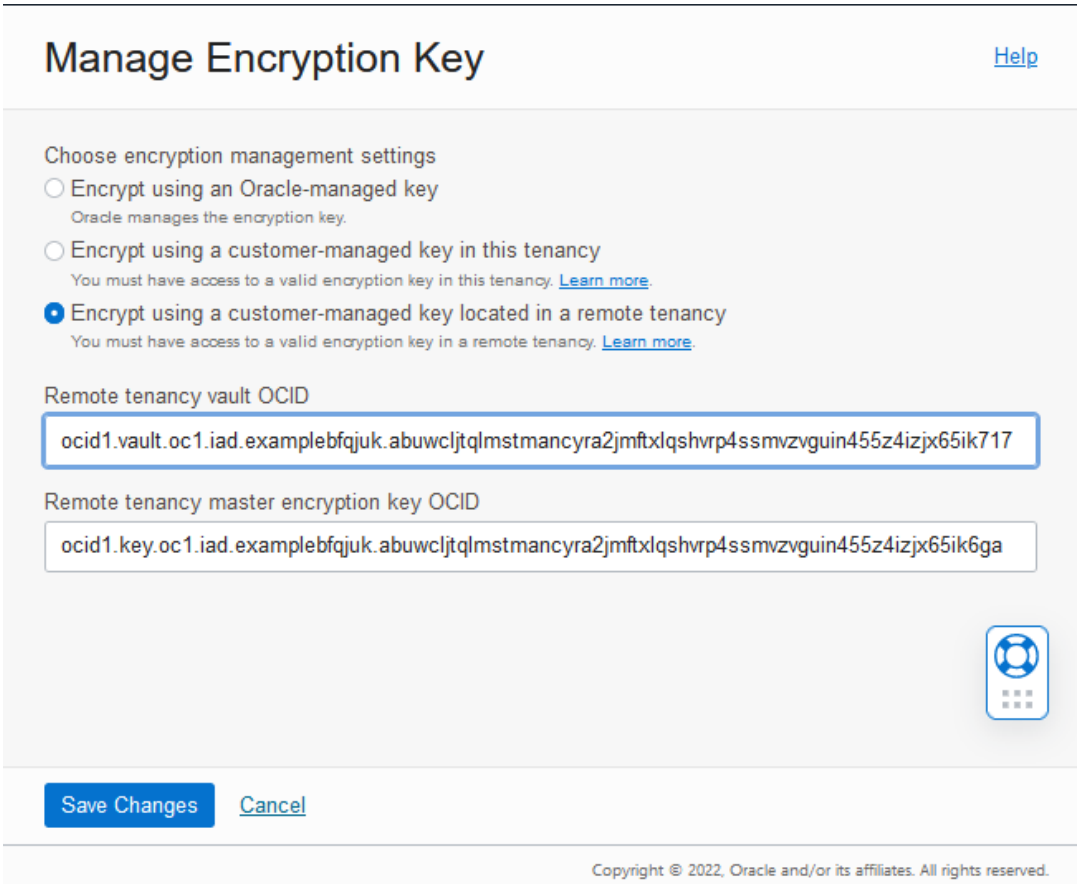
1. Perform the required customer-managed encryption key prerequisite steps as necessary. See Prerequisites to Use Customer-Managed Encryption Keys on Autonomous Database for more information.
2. On the **Details** page, from the **More actions** drop-down list, select **Manage encryption key**.

3. On the **Manage encryption key** page, select the **Encrypt using a customer-managed key located in a different tenancy** option.

   If you are already using customer-managed keys and you want to rotate the TDE keys, follow these steps and use a different key OCID with the same vault OCID, or use a new vault OCID and a new key OCID. This lets you use a key that is different from the current master encryption key.

4. Enter a remote tenancy vault OCID.

5. Enter a remote tenancy master encryption key OCID.



6. Click **Save Changes**.

The **Lifecycle State** changes to **Updating**. When the request completes, the **Lifecycle State** shows **Available**.

After the request completes, on the Oracle Cloud Infrastructure Console, the key information shows on the Autonomous Database Information page under the heading **Encryption**. This area shows the **Encryption Key** field with a link to the Master Encryption Key and the **Encryption Key OCID** field with the Master Encryption Key OCID.

Switch to Oracle-Managed Encryption Keys on Autonomous Database

Shows the steps to switch to Oracle-managed master encryption keys on Autonomous Database if you are using customer-managed encryption keys.

If your Autonomous Database is using customer-managed keys and you want to switch to Oracle-managed keys:

1. On the **Details** page, from the **More actions** drop-down list, select **Manage Encryption Key**.
2. On the **Manage Encryption Key** page, select **Encrypt using Oracle-managed keys**.
3. Click **Save Changes**.

The **Lifecycle State** changes to **Updating**. When the request completes, the **Lifecycle State** shows **Available**.

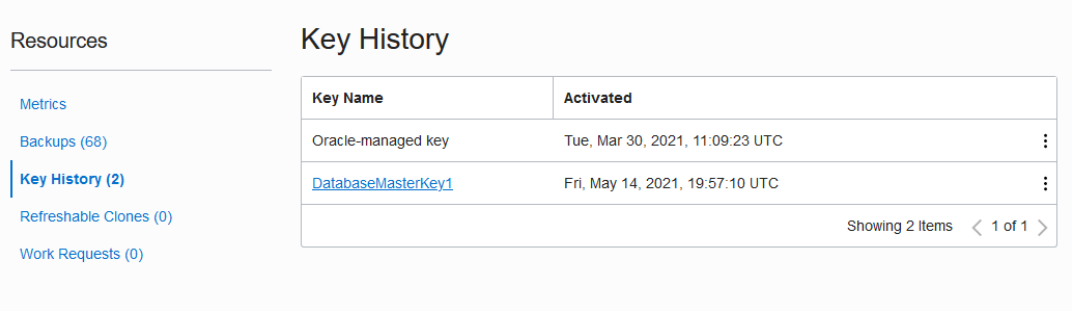
View History for Customer-Managed Encryption Keys on Autonomous Database

You can view the key history from the Oracle Cloud Infrastructure Console or by selecting from the `V$ENCRYPTION_KEYS` view.

To view the key history from the Oracle Cloud Infrastructure Console:

1. On the Autonomous Database Details page, under **Resources**, click **Key History**.
2. This shows the Key History.

   For example:



To view the key history by selecting from the `V$ENCRYPTION_KEYS` view:

1. Connect to the Autonomous Database as the ADMIN user.
2. As the ADMIN user, issue the following command:

```
SELECT activation_time, tag FROM V$ENCRYPTION_KEYS;
```

   For example:

```
SELECT activation_time, tag FROM V$ENCRYPTION_KEYS;
ACTIVATION_TIME                TAG
-----------------------        -----------------------
2020-02-19T17:21:57.821Z       {"credential_name":"OCI$RESOURCE_PRINCIPAL",
                                "oci_management_url":"https://aqm-
management.kms.ca-toronto-1.oraclecloud.com/20180608/keys/",
                                "master_key_id":"ocid1.key.oc1.ca-
toronto-1","username":"\"ADMIN\"",
                                "vault_id":"ocid1.vault.oc1.ca-toronto-1"}
```

See Notes for Using Customer-Managed Keys with Autonomous Database for more information.

Notes for Using Customer-Managed Keys with Autonomous Database

Provides additional information and notes for using customer-managed keys with Autonomous Database

- Caution for Customer-Managed Encryption Keys when Oracle Cloud Infrastructure Vault is Unreachable
  After you switch to customer-managed keys, some database operations might be affected when Oracle Cloud Infrastructure Vault is unreachable, as follows:
- Caution for Using Customer-Managed Encryption Keys When the Master Key is Disabled or Deleted
  After you switch to customer-managed keys, some database operations might be affected if the Oracle Cloud Infrastructure Vault key is disabled or deleted.
- Caution for Using Customer-Managed Encryption Keys History and Backups
  After you switch to customer-managed keys, some database operations might be affected if the master key is rotated, disabled, or deleted and you do not have a valid key to restore your data from a previously saved backup or from a clone.
- Notes for Customer-Managed Keys with Autonomous Data Guard
  Autonomous Data Guard is fully supported when using customer-managed keys with a standby database.
- Notes for Customer-Managed Encryption Keys with Cloning
- Notes for Customer-Managed Keys with Vault in Remote Tenancy
  To use customer-managed master encryption keys with a Vault in a remote tenancy, note the following:

Caution for Customer-Managed Encryption Keys when Oracle Cloud Infrastructure Vault is Unreachable

After you switch to customer-managed keys, some database operations might be affected when Oracle Cloud Infrastructure Vault is unreachable, as follows:

If the database is unable to access Oracle Cloud Infrastructure Vault for some reason, such as a network outage, then Autonomous Database handles the outage as follows:

- There is a 2-hour grace period where the database remains up and running.
- If Oracle Cloud Infrastructure Vault is not reachable at the end of the 2-hour grace period, the database Lifecycle State is set to **Inaccessible**. In this state existing connections are dropped and new connections are not allowed.
- If Autonomous Data Guard is enabled, during or after the 2-hour grace period you can manually try to perform a failover operation. Autonomous Data Guard automatic failover is not triggered when you are using customer-managed encryption keys and the Oracle Cloud Infrastructure Vault is unreachable.
- If Autonomous Database is stopped, then you cannot start the database when the Oracle Cloud Infrastructure Vault is unreachable.

  For this case, the work request shown when you click **Work Requests** on the Oracle Cloud Infrastructure console under Resources shows:

```
The Vault service is not accessible.
Your Autonomous Database could not be started. Please contact Oracle
Support.
```

Caution for Using Customer-Managed Encryption Keys When the Master Key is Disabled or Deleted

After you switch to customer-managed keys, some database operations might be affected if the Oracle Cloud Infrastructure Vault key is disabled or deleted.

- For disable/delete key operations where the Oracle Cloud Infrastructure Vault Master Encryption Key **State** is any of the following:
  - DISABLING
  - DISABLED
  - DELETING
  - DELETED
  - SCHEDULING_DELETION
  - PENDING_DELETION

  The database becomes **Inaccessible** after the Oracle Cloud Infrastructure Vault key lifecycleState goes into one of these states. When the Oracle Cloud Infrastructure Vault key is in any of these states, Autonomous Database drops existing connections and does not allow new connections.

  The database becoming inaccessible can take up to 15min after Oracle Cloud Infrastructure Vault key operations (Autonomous Database checks Oracle Cloud Infrastructure Vault at 15 minute intervals).
- If you disable or delete the Oracle Cloud Infrastructure Vault key used by your Autonomous Database while Autonomous Data Guard is enabled, Autonomous Data Guard will not perform an automatic failover.
- If you enable a disabled key, the database automatically goes into the Available state.
- If you delete the master key you can check the key history in the Oracle Cloud Infrastructure Console to find the keys that were used for the database. The history shows you which Oracle Cloud Infrastructure Vault key OCIDs were used, along with an activation timestamp. If one of the older keys from the history list is still available in the Vault, then you can restore the database to the time when the database was using that key, or you can clone from a backup to create a new database with that timestamp.

See View History for Customer-Managed Encryption Keys on Autonomous Database for more information.

Caution for Using Customer-Managed Encryption Keys History and Backups

After you switch to customer-managed keys, some database operations might be affected if the master key is rotated, disabled, or deleted and you do not have a valid key to restore your data from a previously saved backup or from a clone.

- It is recommended that you create a new Vault key that hasn't been used for rotation in the last 60 days and use that for key rotation. This makes sure that you can go back to a backup if the current Vault key is deleted or disabled.
- When you perform multiple encryption key rotations within 60 days, it is recommended that you either use Oracle Cloud Infrastructure Vault to create a new key for each master encryption key rotation operation or specify a vault key OCID that has not been used in the last 60 days. This helps to save at least one vault key that you can use to recover your data from a backup, in the case where a customer-managed master encryption key is disabled or deleted.

See View History for Customer-Managed Encryption Keys on Autonomous Database for more information.

Notes for Customer-Managed Keys with Autonomous Data Guard

Autonomous Data Guard is fully supported when using customer-managed keys with a standby database.

> **Note:**
>
> When you are using Oracle-managed keys you can only switch to customer-managed keys from the primary database. Likewise, when you are using customer-managed keys you can only rotate keys or switch to Oracle-managed keys from the primary database. The **Manage encryption key** option under **More actions** is disabled for a standby database.

Note the following when you are using Autonomous Data Guard with a standby database and customer-managed keys:

- In order for a remote standby to use the same master encryption key as primary, the master encryption key must be replicated to the remote region. Replication of vaults across regions is only possible if you select the **virtual private vault** option when you create the vault.

  See the following for more information:

  – Autonomous Data Guard with Customer Managed Keys

  – Overview of Vault
- The primary database and the standby database use the same key. In the event of a switchover or failover to the standby, you can continue to use the same key.
- When you rotate the customer-managed key on the primary database, this is reflected on the standby database.
- When you switch from customer-managed keys to Oracle-managed keys the change is reflected on the standby database.
- When the Oracle Cloud Infrastructure Vault is unreachable, there is a 2-hour grace period before the database goes into the `INACCESSIBLE` state. You can perform a failover during or after this 2-hour grace period.
- If you disable or delete the Oracle Cloud Infrastructure master encryption key that your Autonomous Database is using with customer-managed keys while Autonomous Data Guard is enabled, Autonomous Data Guard does not perform an automatic failover.

Notes for Customer-Managed Encryption Keys with Cloning

- Autonomous Database does not support using customer-managed encryption keys with a refreshable clone. You cannot create a refreshable clone from a source database that uses customer-managed encryption keys. Additionally, you cannot switch to a customer-managed encryption key on a source database that has one or more refreshable clones.

Notes for Customer-Managed Keys with Vault in Remote Tenancy

To use customer-managed master encryption keys with a Vault in a remote tenancy, note the following:

When you use customer-managed master encryption keys with a Vault in a remote tenancy, the Vault and the Autonomous Database instance must be in the same region. To change the tenancy, on the sign-on page click **Change tenancy**. After you change the tenancy, make sure to select the same region for both the Vault and the Autonomous Database instance.

Configure Network Access with Access Control Rules (ACLs) and Private Endpoints

Provides details on how to configure network access with access control rules or using a private endpoint and describes the secure client connection options. Also describes how to enable support for TLS connections (require mutual TLS only or allow both mutual TLS and TLS authentication).

- Configuring Network Access with Access Control Rules (ACLs)
  Specifying an access control list blocks all IP addresses that are not in the ACL list from accessing the database. After you specify an access control list, the Autonomous Database only accepts connections from addresses on the access control list and the database rejects all other client connections.
- Configure Network Access with Private Endpoints
  You can specify that Autonomous Database uses a private endpoint inside your Virtual Cloud Network (VCN) in your tenancy. You can configure a private endpoint during provisioning or cloning your Autonomous Database, or you can switch to using a private endpoint in an existing database that uses a public endpoint. This allows you to keep all traffic to and from your database off of the public internet.
- Update Network Options to Allow TLS or Require Only Mutual TLS (mTLS) Authentication on Autonomous Database
  Describes how to update the secure client connection authentication options, Mutual TLS (mTLS) and TLS.

Configuring Network Access with Access Control Rules (ACLs)

Specifying an access control list blocks all IP addresses that are not in the ACL list from accessing the database. After you specify an access control list, the Autonomous Database only accepts connections from addresses on the access control list and the database rejects all other client connections.

- Configure Access Control Lists When You Provision or Clone an Instance
  When you provision or clone Autonomous Database with the **Secure access from allowed IPs and VCNs only** option, you can restrict network access by defining Access Control Lists (ACLs).
- Configure Access Control Lists for an Existing Autonomous Database Instance
  You can control and restrict access to your Autonomous Database by specifying network access control lists (ACLs). On an existing Autonomous Database instance with a public endpoint you can add, change, or remove ACLs.
- Change from Private to Public Endpoints with Autonomous Database
  If your Autonomous Database instance is configured to use a private endpoint you can change the configuration to use a public endpoint.
- Access Control List Restrictions and Notes
  Describes restrictions and notes for access control rules on Autonomous Database.

Configure Access Control Lists When You Provision or Clone an Instance

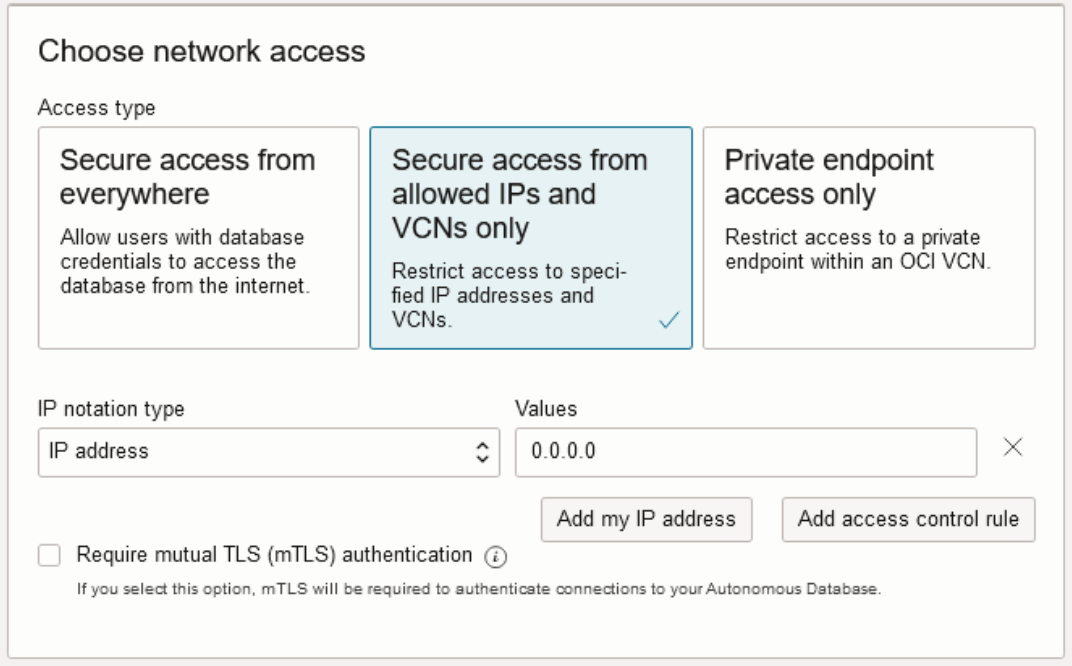
When you provision or clone Autonomous Database with the **Secure access from allowed IPs and VCNs only** option, you can restrict network access by defining Access Control Lists (ACLs).

See Provision Autonomous Database for information on provisioning your Autonomous Database.

Configure ACLs as follows:

1. In the Choose network access area, select **Secure access from allowed IPs and VCNs only**.

   With **Secure access from allowed IPs and VCNs only** selected, the console shows the fields and options to specify ACLs:



2. In the Choose network access area, specify access control rules by selecting an **IP notation type** and entering **Values** appropriate for the type you select:

   - **IP address**:
     In **Values** field enter values for the **IP address**. An IP address specified in a network ACL entry is the public IP address of the client that is visible on the public internet that you want to grant access. For example, for an Oracle Cloud Infrastructure VM, this is the IP address shown in the **Public IP** field on the Oracle Cloud Infrastructure console for that VM.

     > **Note:**
     >
     > Optionally click **Add my IP address** to add your current IP address to the ACL entry.

   - **CIDR block**:

In **Values** field enter values for the **CIDR block**. The CIDR block specified is the public CIDR block of the clients that are visible on the public internet that you want to grant access.

- **Virtual cloud network**:
  Use this option to specify the VCN for use with an Oracle Cloud Infrastructure Service Gateway. See Access to Oracle Services: Service Gateway for more information.
  - In **Virtual cloud network** field select the VCN that you want to grant access from. If you do not have the privileges to see the VCNs in your tenancy this list is empty. In this case use the selection **Virtual cloud network (OCID)** to specify the OCID of the VCN.
  - Optionally, in the **IP addresses or CIDRs** field enter private IP addresses or private CIDR blocks as a comma separated list to allow specific clients in the VCN.
- **Virtual cloud network (OCID)**:
  Use this option to specify the VCN (OCID) for use with an Oracle Cloud Infrastructure Service Gateway. See Access to Oracle Services: Service Gateway for more information.
  - In the **Values** field enter the OCID of the VCN you want to grant access from.
  - Optionally, in the **IP addresses or CIDRs** field enter private IP addresses or private CIDR blocks as a comma separated list to allow specific clients in the VCN.

If you want to specify multiple IP addresses or CIDR ranges within the same VCN, do not create multiple ACL entries. Use one ACL entry with the values for the multiple IP addresses or CIDR ranges separated by commas.

3. Click **Add access control rule** to add a new value to the access control list.
4. Click **x** to remove an entry.

   You can also clear the value in the **IP addresses** or **CIDR blocks** field to remove an entry.
5. Require mutual TLS (mTLS) authentication.

   After you enter an IP notation type and a value, you have the option to select this option. The options are:

   - When **Require mutual TLS (mTLS) authentication** is selected, only mTLS connections are allowed (TLS authentication is not allowed).
   - When **Require mutual TLS (mTLS) authentication** is deselected, TLS and mTLS connections are allowed. This is the default configuration.

   See Update Network Options to Allow TLS or Require Only Mutual TLS (mTLS) Authentication on Autonomous Database for more information.
6. Complete the remaining provisioning or cloning steps, as specified in Provision Autonomous Database, Clone an Autonomous Database Instance, or Clone an Autonomous Database from a Backup.

After provisioning completes, you can update public endpoint ACLs or you can change the Autonomous Database configuration to use a private endpoint.

See Configure Access Control Lists for an Existing Autonomous Database Instance for information on updating ACLs.

See Change from Public to Private Endpoints with Autonomous Database for information on changing to a private endpoint.

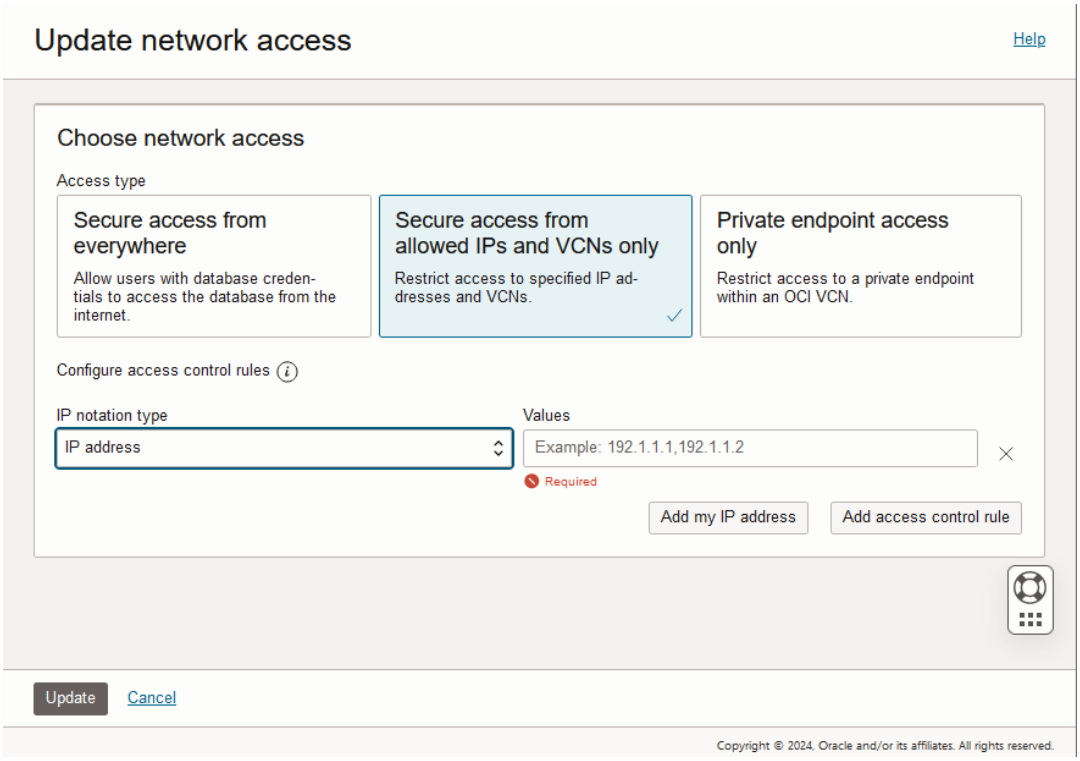
Configure Access Control Lists for an Existing Autonomous Database Instance

You can control and restrict access to your Autonomous Database by specifying network access control lists (ACLs). On an existing Autonomous Database instance with a public endpoint you can add, change, or remove ACLs.

Configure ACLs, or add, remove, or update existing ACLs for an Autonomous Database instance as follows:

1. On the **Details** page, from the **More actions** drop-down list, select **Update network access**.

   This shows the Update network access dialog.



2. In the dialog, under **Access type**, select **Secure access from allowed IPs and VCNs only** and specify the access control rules by selecting an **IP notation type** and values:

   Select one of:

   - **IP address**:
     In **Values** field enter values for the **IP address**. An IP address specified in a network ACL entry is the public IP address of the client that is visible on the public internet that you want to grant access. For example, for an Oracle Cloud Infrastructure VM, this is the IP address shown in the **Public IP** field on the Oracle Cloud Infrastructure console for that VM.

     > **Note:**
     >
     > Optionally click **Add my IP address** to add your current IP address to the ACL entry.

   - **CIDR block**:

In **Values** field enter values for the **CIDR block**. The CIDR block specified is the public CIDR block of the clients that are visible on the public internet that you want to grant access.

- **Virtual cloud network**:
  Use this option to specify the VCN for use with an Oracle Cloud Infrastructure Service Gateway. See Access to Oracle Services: Service Gateway for more information.
  - In **Virtual cloud network** field select the VCN that you want to grant access from. If you do not have the privileges to see the VCNs in your tenancy this list is empty. In this case use the selection **Virtual cloud network (OCID)** to specify the OCID of the VCN.
  - Optionally, in the **IP addresses or CIDRs** field enter private IP addresses or private CIDR blocks as a comma separated list to allow specific clients in the VCN.
- **Virtual cloud network (OCID)**:
  Use this option to specify the VCN for use with an Oracle Cloud Infrastructure Service Gateway. See Access to Oracle Services: Service Gateway for more information.
  - In the **Values** field enter the OCID of the VCN you want to grant access from.
  - Optionally, in the **IP addresses or CIDRs** field enter private IP addresses or private CIDR blocks as a comma separated list to allow specific clients in the VCN.

If you want to specify multiple IP addresses or CIDR ranges within the same VCN, do not create multiple ACL entries. Use one ACL entry with the values for the multiple IP addresses or CIDR ranges separated by commas.

3. Click **Add access control** to add a new value to the access control list.
4. Click **x** to remove an entry.

   You can also clear the value in the **IP addresses** or **CIDR blocks** field to remove an entry.
5. Click **Update**.

If the Lifecycle State is **Available** when you click **Update** the Lifecycle State changes to **Updating** until the ACL is set. The database is still up and accessible, there is no downtime. When the update is complete the Lifecycle State returns to **Available** and the network ACLs from the access control list are in effect.

Change from Private to Public Endpoints with Autonomous Database

If your Autonomous Database instance is configured to use a private endpoint you can change the configuration to use a public endpoint.

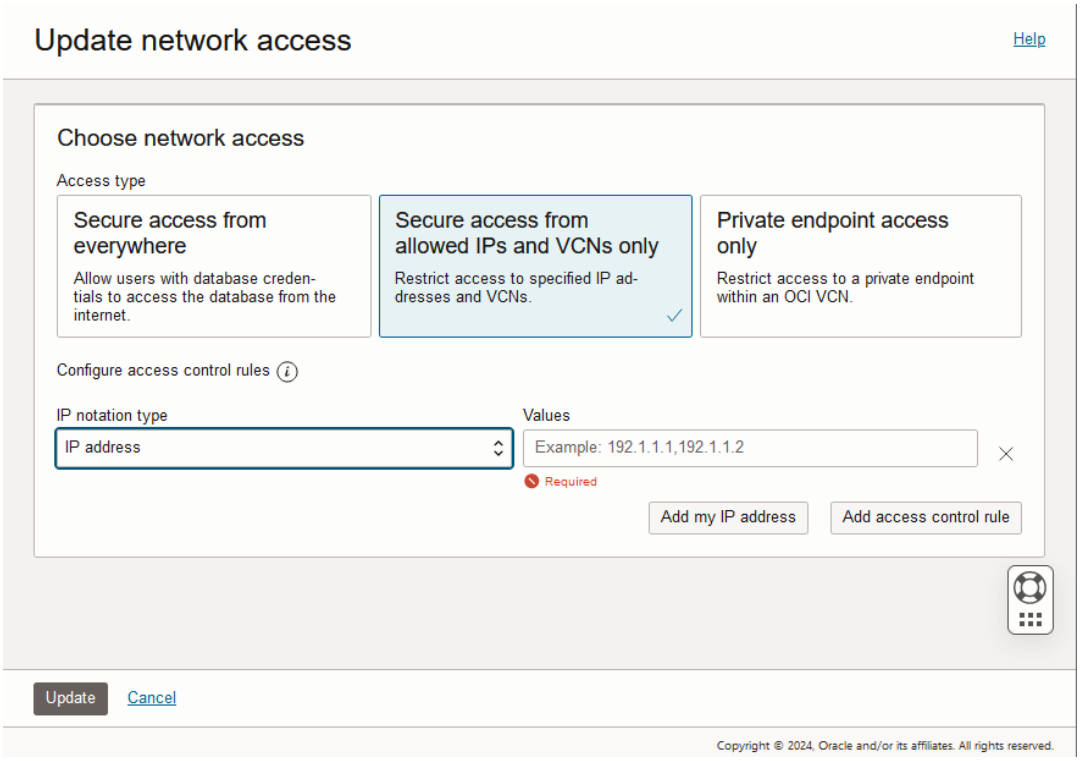
There are several prerequisites to change an instance from a private to a public endpoint, as follows:

- The Autonomous Database instance must be in the Available state (Lifecycle State: Available).
- Before changing the network configuration from a private endpoint to a public endpoint, you must change the configuration to not allow TLS connections. This closes any existing TLS connections. See Update your Autonomous Database Instance to Require mTLS and Disallow TLS Authentication for more information.

To specify a public endpoint for your Autonomous Database do the following:

1. On the **Details** page, from the **More actions** drop-down list, select **Update network access**.
2. In the **Update network access** dialog, select one of **Secure access from everywhere** or **Secure access from allowed IPs and VCNs only**.

For example, if you select **Secure access from allowed IPs and VCNs only** the dialog shows fields to configure access control rules:



3. In the dialog, under Configure access control rules specify rules by selecting an **IP notation type** and values:

   - **IP address**:
     In **Values** field enter values for the **IP address**. An IP address specified in a network ACL entry is the public IP address of the client that is visible on the public internet that you want to grant access. For example, for an Oracle Cloud Infrastructure VM, this is the IP address shown in the **Public IP** field on the Oracle Cloud Infrastructure console for that VM.

     > **Note:**
     >
     > Optionally click **Add my IP address** to add your current IP address to the ACL entry.

   - **CIDR block**:
     In **Values** field enter values for the **CIDR block**. The CIDR block specified is the public CIDR block of the clients that are visible on the public internet that you want to grant access.

   - **Virtual cloud network**:
     - In **Virtual cloud network** field select the VCN that you want to grant access from. If you do not have the privileges to see the VCNs in your tenancy this list is empty. In this case use the selection **Virtual cloud network (OCID)** to specify the OCID of the VCN.
     - Optionally, in the **IP addresses or CIDRs** field enter private IP addresses or private CIDR blocks as a comma separated list to allow specific clients in the VCN.

- **Virtual cloud network (OCID)**:
  - In the **Values** field enter the OCID of the VCN you want to grant access from.
  - Optionally, in the **IP addresses or CIDRs** field enter private IP addresses or private CIDR blocks as a comma separated list to allow specific clients in the VCN.

  If you want to specify multiple IP addresses or CIDR ranges within the same VCN, do not create multiple ACL entries. Use one ACL entry with the values for the multiple IP addresses or CIDR ranges separated by commas.

4. Click **Add access control rule** to add a new value to the access control list.
5. Click **x** to remove an entry.

   You can also clear the value in the **IP addresses** or **CIDR blocks** field to remove an entry.
6. Click **Update**.
7. In the **Confirm** dialog, type the Autonomous Database name to confirm the change.
8. In the **Confirm** dialog, click **Update**.

The Lifecycle State changes to **Updating** until the operation completes.

Notes for changing from private endpoint to public endpoint network access:

- After updating the network access type all database users must obtain a new wallet and use the new wallet to access the database. See Download Client Credentials (Wallets) for more information.
- After the update completes, you can change or define new access control rules ACLs for the public endpoint. See Configure Access Control Lists for an Existing Autonomous Database Instance for more information.
- The URL for Database Actions and for the Database Tools are different when a database uses a private endpoint compared to using a public endpoint. Click Database Actions on the Oracle Cloud Infrastructure Console to find the updated Database Actions URL and in Database Actions click the appropriate cards to find the updated Database Tools URLs, after changing from a private endpoint to a public endpoint.

Access Control List Restrictions and Notes

Describes restrictions and notes for access control rules on Autonomous Database.

- If you want to only allow connections coming through a service gateway you need to use the IP address of the service gateway in your ACL definition. To do this you need to add an ACL definition with the CIDR source type with the value `240.0.0.0/4`. Note that this is not recommended, instead of this you can specify individual VCNs in your ACL definition for the VCNs you want to allow access from.

  See Access to Oracle Services: Service Gateway for more information.
- When you restore a database the existing ACLs are not overwritten by the restore.
- The network ACLs apply to the database connections and Oracle Machine Learning notebooks. If an ACL is defined, if you try to login to Oracle Machine Learning Notebooks from a client whose IP is not specified on the ACL this shows the "login rejected based on access control list set by the administrator" error.
- The following Autonomous Database tools are subject to ACLs. You can use Virtual Cloud Network, Virtual Cloud Network (OCID), IP address, or CIDR block ACLs to control access to these tools:
  - Database Actions

  - Oracle APEX
  - Oracle Graph Studio
  - Oracle Machine Learning Notebooks
  - Oracle REST Data Services
- If you have a private subnet in your VCN that is configured to access the public internet through a NAT Gateway, you need to enter the public IP address of the NAT Gateway in your ACL definition. Clients in the private subnet do not have public IP addresses. See NAT Gateway for more information.
- If you are using ACLs and TLS connections are allowed, you must change your network configuration to not allow TLS connections before removing all ACLs. See Update your Autonomous Database Instance to Require mTLS and Disallow TLS Authentication for more information.

Configure Network Access with Private Endpoints

You can specify that Autonomous Database uses a private endpoint inside your Virtual Cloud Network (VCN) in your tenancy. You can configure a private endpoint during provisioning or cloning your Autonomous Database, or you can switch to using a private endpoint in an existing database that uses a public endpoint. This allows you to keep all traffic to and from your database off of the public internet.

Specifying the virtual cloud network configuration allows traffic only from the virtual cloud network you specify and blocks access to the database from all public IPs or VCNs. This allows you to define security rules with Security Lists or at the Network Security Group (NSG) level to specify ingress/egress for your Autonomous Database instance. Using a private endpoint and defining Security Lists or NSGs allows you to control traffic to and from your Autonomous Database instance.

- Configure Private Endpoints
  You can specify that Autonomous Database uses a private endpoint and configure a Virtual Cloud Network (VCN) in your tenancy to use with the private endpoint.
- Enhanced Security for Outbound Connections with Private Endpoints
  When you define a private endpoint for your Autonomous Database instance you can provide enhanced security by setting a database property to enforce that all outgoing connections to a target host are subject to and limited by the private endpoint's egress rules. You define egress rules in the Virtual Cloud Network (VCN) security list or in the Network Security Group (NSG) associated with the Autonomous Database instance private endpoint.
- Private Endpoints Notes
  Describes restrictions and notes for private endpoints on Autonomous Database.
- Private Endpoints Configuration Examples on Autonomous Database
  Shows several Private Endpoint (VCN) configuration samples for Autonomous Database.

Configure Private Endpoints

You can specify that Autonomous Database uses a private endpoint and configure a Virtual Cloud Network (VCN) in your tenancy to use with the private endpoint.

Perform the following prerequisite steps before configuring a private endpoint:

- Set required policies for the resources you are working with. See Prerequisite: IAM Policies Required to Manage Private Endpoints for more information.

- Create a VCN within the region that will contain your Autonomous Database. See VCNs and Subnets for more information.
- Configure a subnet within your VCN configured with default DHCP options. See DNS in Your Virtual Cloud Network for more information.
- (Optional) Perform the following optional step before configuring a private endpoint:

  Specify a Network Security Group (NSG) within your VCN. The NSG specifies rules for connections to your Autonomous Database. See Network Security Groups for more information.

You can configure the private endpoint for an existing Autonomous Database instance or when you provision or clone a new instance:

- Prerequisite: IAM Policies Required to Manage Private Endpoints
  Autonomous Database relies on the IAM (Identity and Access Management) service to authenticate and authorize cloud users to perform operations that use any of the Oracle Cloud Infrastructure interfaces (the Console, REST API, CLI, SDK, or others).
- Configure Private Endpoints When You Provision or Clone an Instance
  You can configure a private endpoint when you provision or clone an Autonomous Database instance.
- Change from Public to Private Endpoints with Autonomous Database
  If your Autonomous Database instance is configured to use a public endpoint you can change the configuration to a private endpoint.

Prerequisite: IAM Policies Required to Manage Private Endpoints

Autonomous Database relies on the IAM (Identity and Access Management) service to authenticate and authorize cloud users to perform operations that use any of the Oracle Cloud Infrastructure interfaces (the Console, REST API, CLI, SDK, or others).

The IAM service uses **groups**, **compartments** and **policies** to control which cloud users can access which resources. In particular, a policy defines what kind of access a group of users has to a particular kind of resource in a particular compartment. For more information, see Getting Started with Policies.

In addition to the policies required to provision and manage an Autonomous Database, some network policies are needed to use private endpoints. The following table lists the IAM policies required for a cloud user to add a private endpoint.

> **Note:**
>
> The listed policies are the minimum requirements to add a private endpoint. You can also use a policy rule that is broader. For example, if you set the policy rule:
>
> ```
> Allow group MyGroupName to manage virtual-network-family in tenancy
> ```
>
> This rule also works because it is a superset that contains all the required policies.

Operation	Required IAM Policies
Configure a private endpoint	`use vcns` for the compartment which the VCN is in `use subnets` for the compartment which the VCN is in `use network-security-groups` for the compartment which the network security group is in `manage private-ips` for the compartment which the VCN is in `manage vnics` for the compartment which the VCN is in `manage vnics` for the compartment which the database is provisioned or is to be provisioned in

See Common Policies for more information.

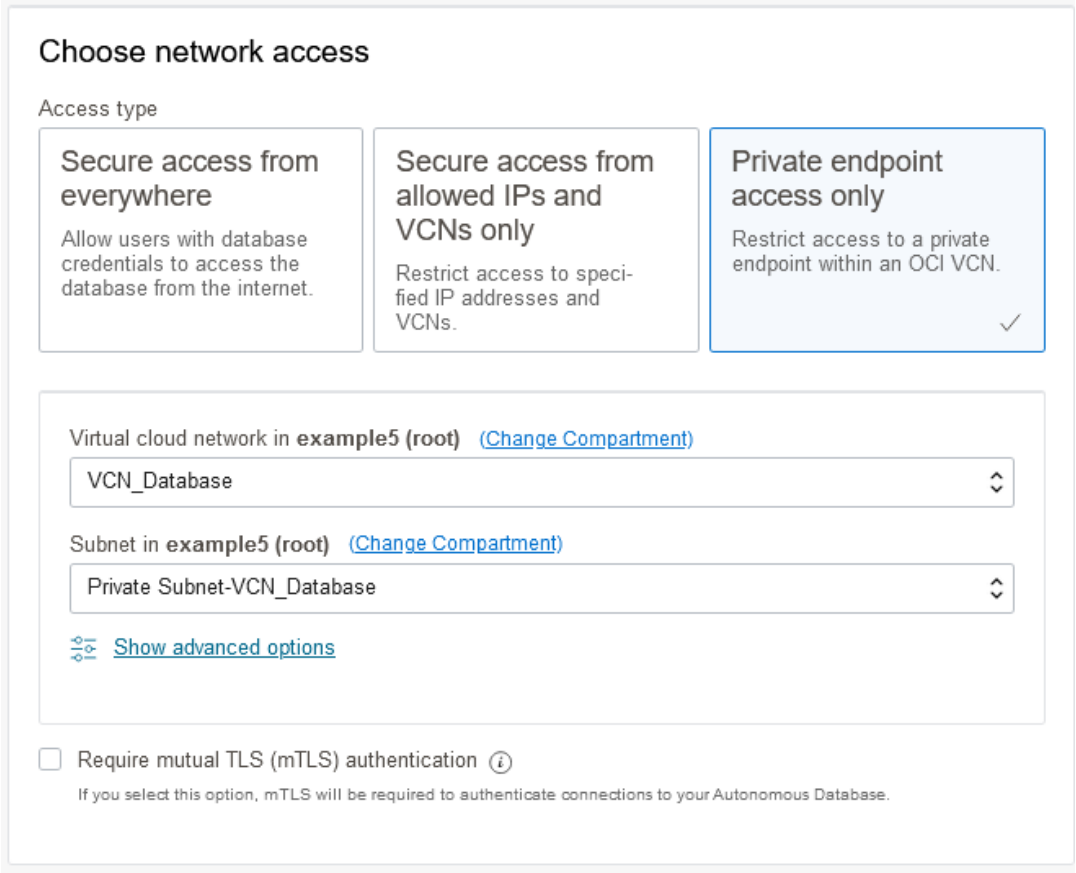
Configure Private Endpoints When You Provision or Clone an Instance

You can configure a private endpoint when you provision or clone an Autonomous Database instance.

These steps assume you are provisioning or cloning an instance and you have completed the prerequisite steps, and you are at the **Choose network access** step of the provisioning or cloning steps:

1. Select **Private endpoint access only**.

   This expands the Virtual cloud network private access configuration area.



> **Note:**
>
> If you select **Private endpoint access only**, this only allows connections from the specified private network (VCN), from peered VCNs, and from on-prem networks connected to your VCN. You can configure an Autonomous Database instance on a private endpoint to allow connections from on-prem networks. See Example: Connecting from Your Data Center to Autonomous Database for an example.

If you want to allow connections from public IP addresses, then you need to select either **Secure access from everywhere** or **Secure access from allowed IPs and VCNs only** when you provision or clone your Autonomous Database.

2. Select a **Virtual cloud network** in your compartment or if the VCN is in a different compartment click **Change Compartment** and select the compartment that contains the VCN and then select a virtual cloud network.

   See VCNs and Subnets for more information.

3. Select the **Subnet** in your compartment to attach the Autonomous Database to or if the Subnet is in a different compartment click **Change Compartment** and select the compartment that contains the Subnet and then select a subnet.

   See VCNs and Subnets for more information.

4. (Optional) Click **Show advanced options** to configure additional private endpoint options.

   a. Optionally enter a **Private IP address**.

      Use this field to enter a custom private IP address. The private IP address you enter must be within the selected subnet's CIDR range.

      If you do not provide a custom private IP address the IP address is automatically assigned.

   b. Optionally enter a **Hostname prefix**.

      This specifies a hostname prefix for the Autonomous Database and associates a DNS name with the database instance, in the following form:

      ```
      hostname_prefix.adb.region.oraclecloud.com
      ```

      If you do not specify a hostname prefix, a system generated hostname prefix is supplied.

   c. Optionally add **Network security groups (NSGs)**.

      Optionally, to allow connections to the Autonomous Database instance define security rules in an NSG; this creates a virtual firewall for your Autonomous Database.

      - Select a Network Security Group in your compartment to attach the Autonomous Database to, or if the Network Security Group is in a different compartment, click **Change Compartment** and select a different compartment and then select a Network Security Group in that compartment.
      - Click **+ Another Network Security Group** to add another Network Security Group.
      - Click **x** to remove a Network Security Group entry.

      For the NSG you select for the private endpoint define a security rule as follows:

- For mutual TLS (mTLS) authentication, add a stateful ingress rule with the source set to the address range you want to allow to connect to your database, the IP Protocol set to TCP, and the Destination Port Range set to 1522. See About Mutual TLS (mTLS) Authentication for more information.
- For TLS authentication, add a stateful ingress rule with the source set to the address range you want to allow to connect to your database, the IP Protocol set to TCP, and the Destination Port Range set to 1521. See About TLS Authentication for more information.
- To use Oracle APEX, Database Actions, and Oracle REST Data Services, add port 443 to the NSG rule.

> **Note:**
>
> Incoming and outgoing connections are limited by the combination of ingress and egress rules defined in NSGs and the Security Lists defined with the VCN. When there are no NSGs, ingress and egress rules defined in the Security Lists for the VCN still apply. See Security Lists for more information on working with Security Lists.

See Private Endpoints Configuration Examples on Autonomous Database for examples.

See Network Security Groups for more information.

5. Require mutual TLS (mTLS) authentication.

   The **Require mutual TLS (mTLS) authentication** options are:

   - When **Require mutual TLS (mTLS) authentication** is deselected, TLS and mTLS connections are allowed. This is the default configuration.
   - When **Require mutual TLS (mTLS) authentication** is selected, only mTLS connections are allowed (TLS authentication is not allowed).

   See Update Network Options to Allow TLS or Require Only Mutual TLS (mTLS) Authentication on Autonomous Database for more information.

6. Complete the remaining provisioning or cloning steps, as specified in Provision Autonomous Database, Clone an Autonomous Database Instance, or Clone an Autonomous Database from a Backup.

See Private Endpoints Notes for more information.

Change from Public to Private Endpoints with Autonomous Database

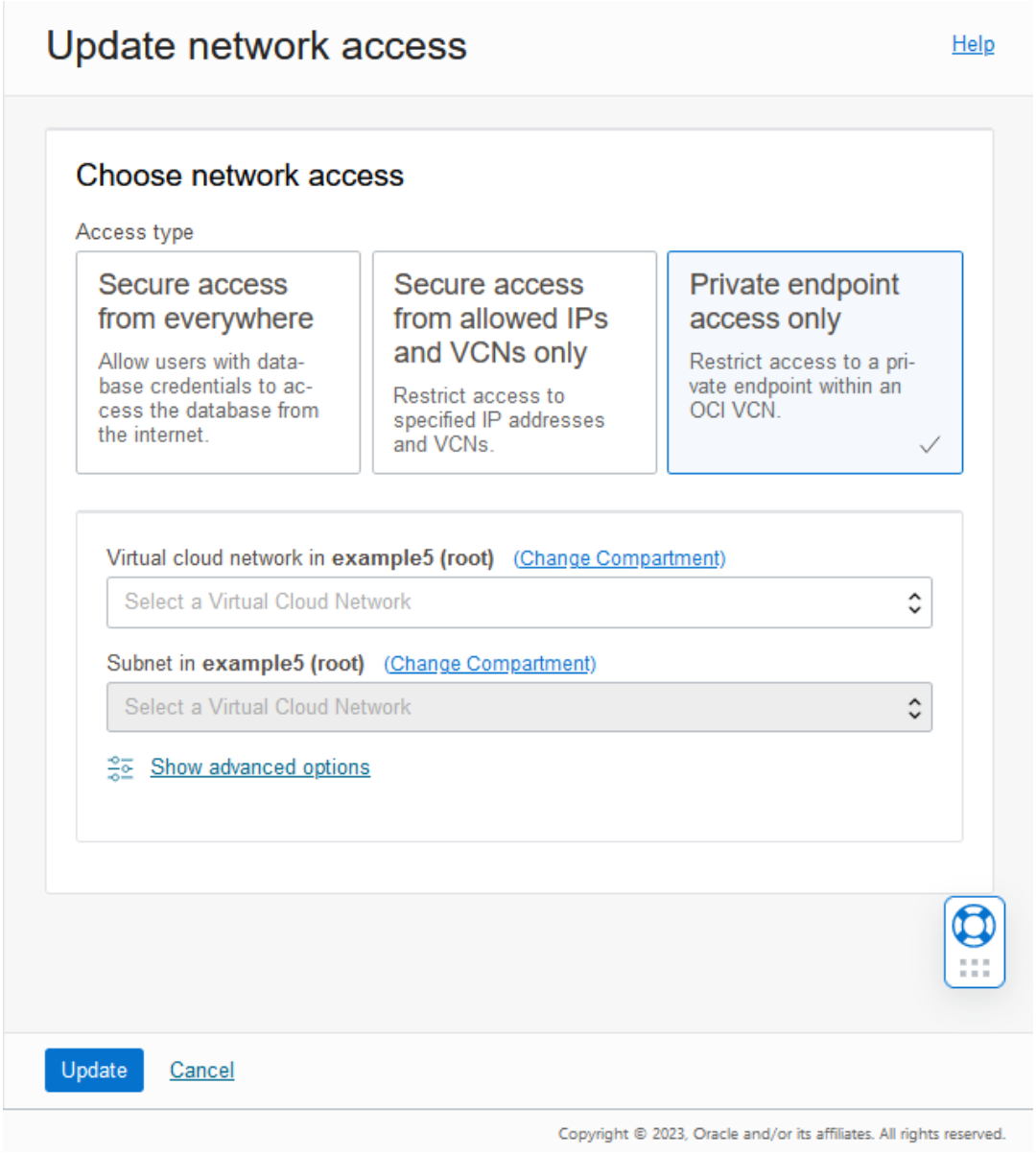
If your Autonomous Database instance is configured to use a public endpoint you can change the configuration to a private endpoint.

1. On the **Details** page, from the **More actions** drop-down list, select **Update network access**.

   To change an instance from a public to a private endpoint, the Autonomous Database instance must be in the **Available** state (Lifecycle State: **Available**).

2. In the **Update network access** dialog, select **Private endpoint access only**.

   This expands the Virtual cloud network private access configuration area.



> **Note:**
>
> If you select **Private endpoint access only**, this only allows connections from the specified private network (VCN), from peered VCNs, and from on-prem networks connected to your VCN. Thus, you can configure an Autonomous Database instance on a private endpoint to allow connections from on-prem networks. See Example: Connecting from Your Data Center to Autonomous Database for an example.

3. (Optional) Click **Show advanced options** for additional options.
   a. Optionally enter a **Private IP address**.

      Use this field to enter a custom private IP address. The private IP address you enter must be within the selected subnet's CIDR range.

      If you do not provide a custom private IP address the IP address is automatically assigned.

b. Optionally enter a **Hostname prefix**.

This specifies a hostname prefix for the Autonomous Database and associates a DNS name with the database instance, in the following form:

```
hostname_prefix.adb.region.oraclecloud.com
```

If you do not specify a hostname prefix, a system generated hostname prefix is supplied.

c. Optionally add **Network security groups (NSGs)**.

Optionally, to allow connections to the Autonomous Database instance define security rules in an NSG; this creates a virtual firewall for your Autonomous Database.

- Select a Network Security Group in your compartment to attach the Autonomous Database to, or if the Network Security Group is in a different compartment, click **Change Compartment** and select a different compartment and then select a Network Security Group in that compartment.
- Click **+ Another Network Security Group** to add another Network Security Group.
- Click **x** to remove a Network Security Group entry.

For the NSG you select for the private endpoint define a security rule as follows:

- For mutual TLS (mTLS) authentication, add a stateful ingress rule with the source set to the address range you want to allow to connect to your database, the IP Protocol set to TCP, and the Destination Port Range set to 1522. See About Mutual TLS (mTLS) Authentication for more information.
- For TLS authentication, add a stateful ingress rule with the source set to the address range you want to allow to connect to your database, the IP Protocol set to TCP, and the Destination Port Range set to 1521. See About TLS Authentication for more information.
- To use Oracle APEX, Database Actions, and Oracle REST Data Services, add port 443 to the NSG rule.

> **Note:**
>
> Incoming and outgoing connections are limited by the combination of ingress and egress rules defined in NSGs and the Security Lists defined with the VCN. When there are no NSGs, ingress and egress rules defined in the Security Lists for the VCN still apply. See Security Lists for more information on working with Security Lists.

See Private Endpoints Configuration Examples on Autonomous Database for examples.

See Network Security Groups for more information.

4. Click **Update**.
5. In the **Confirm** dialog, type the Autonomous Database name to confirm the change.
6. In the **Confirm** dialog, click **Update**.

The Lifecycle State changes to **Updating** until the operation completes.

Notes for changing from public to private network access:

- After updating the network access type all database users must obtain a new wallet and use the new wallet to access the database. See Download Client Credentials (Wallets) for more information.
- If you had ACLs defined for the public endpoint, the ACLs do not apply for the private endpoint.
- After you update the network access to use a private endpoint, the URL for the Database Tools is different compared to using a public endpoint. You can find the updated URLs on the console, after changing from a public endpoint to a private endpoint.

Enhanced Security for Outbound Connections with Private Endpoints

When you define a private endpoint for your Autonomous Database instance you can provide enhanced security by setting a database property to enforce that all outgoing connections to a target host are subject to and limited by the private endpoint's egress rules. You define egress rules in the Virtual Cloud Network (VCN) security list or in the Network Security Group (NSG) associated with the Autonomous Database instance private endpoint.

Before you set this database property configure your Autonomous Database instance to use a private endpoint. See Configure Private Endpoints for more information.

Set the `ROUTE_OUTBOUND_CONNECTIONS` database property to `PRIVATE_ENDPOINT` to specify that all outgoing connections are subject to the Autonomous Database instance private endpoint VCN's egress rules. With the value `PRIVATE_ENDPOINT` the database restricts outgoing connections to locations specified by the private endpoint's egress rules.

> **Note:**
>
> With `ROUTE_OUTBOUND_CONNECTIONS` not set to `PRIVATE_ENDPOINT`, all outgoing connections to the public internet pass through the Network Address Translation (NAT) Gateway of the service VCN. In this case, if the target host is on a public endpoint the outgoing connections are not subject to the Autonomous Database instance private endpoint VCN or NSG egress rules.

When you configure a private endpoint for your Autonomous Database instance and set `ROUTE_OUTBOUND_CONNECTIONS` to `PRIVATE_ENDPOINT`, this setting changes the handling of outbound connections for the following:

- Database links
- APEX_LDAP, APEX_MAIL, and APEX_WEB_SERVICE
- UTL_HTTP, UTL_SMTP, and UTL_TCP
- DBMS_LDAP
- CMU with Microsoft Active Directory

  See Use Microsoft Active Directory with Autonomous Database for more information.

To set `ROUTE_OUTBOUND_CONNECTIONS`:

1. Connect to your database.
2. Set the database property `ROUTE_OUTBOUND_CONNECTIONS`.

For example:

```
ALTER DATABASE PROPERTY SET ROUTE_OUTBOUND_CONNECTIONS =
'PRIVATE_ENDPOINT';
```

Notes for setting `ROUTE_OUTBOUND_CONNECTIONS`:

- Use the following command to restore the default parameter value:

  ```
  ALTER DATABASE PROPERTY SET ROUTE_OUTBOUND_CONNECTIONS = '';
  ```

- Use the following command to query the current parameter value:

  ```
  SELECT * FROM DATABASE_PROPERTIES
          WHERE PROPERTY_NAME = 'ROUTE_OUTBOUND_CONNECTIONS';
  ```

  If the property is not set the query does not return results.

- This property only applies for database links that you create after you set the property to the value `PRIVATE_ENDPOINT`. Thus, database links that you created prior to setting the property continue to use the NAT Gateway of the service VCN and are not subject to the Autonomous Database instance private endpoint's egress rules.
- Only set `ROUTE_OUTBOUND_CONNECTIONS` to the value `PRIVATE_ENDPOINT` when you are using Autonomous Database with a private endpoint.
- By default, when you are accessing other private endpoints the connection is subject to your VCN's egress rules. Setting `ROUTE_OUTBOUND_CONNECTIONS` has no effect in this case. The `ROUTE_OUTBOUND_CONNECTIONS` property applies when you want outgoing connections to follow the private endpoint egress rules even when accessing public endpoints.

See NAT Gateway for more information on Network Address Translation (NAT) gateway.

Private Endpoints Notes

Describes restrictions and notes for private endpoints on Autonomous Database.

- After you update the network access to use a private endpoint, or after the provisioning or cloning completes where you configure a private endpoint, you can view the network configuration on the Autonomous Database Details page under the **Network** section.

  The **Network** section shows the following information for a private endpoint:

  - **Access Type**: Specifies the access type for the Autonomous Database configuration. Private endpoint configurations show the access type: **Virtual Cloud Network**.
  - **Virtual Cloud Network**: This includes a link for the VCN associated with the private endpoint.
  - **Subnet**: This includes a link for the subnet associated with the private endpoint.
  - **Private IP**: Shows the private IP for the private endpoint configuration.
  - **Network Security Groups**: This field includes links to the NSG(s) configured with the private endpoint.

- After provisioning or cloning completes, you can change the Autonomous Database configuration to use a public endpoint.

  See Change from Private to Public Endpoints with Autonomous Database for information on changing to a public endpoint.

- You can specify up to five NSGs to control access to your Autonomous Database.
- You can change the private endpoint Network Security Group (NSG) for the Autonomous Database.

  To change the NSG for a private endpoint, do the following:

  1. On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.
  2. On the **Autonomous Database Details** page, under **Network** in the **Network Security Groups** field, click **Edit**.
- You can connect your Oracle Analytics Cloud instance to your Autonomous Database that has a private endpoint using the Data Gateway like you do for an on-premises database. See Configure and Register Data Gateway for Data Visualization for more information.
- The following Autonomous Database tools are supported in databases configured with a private endpoint:
  - Database Actions
  - Oracle APEX
  - Oracle Graph Studio
  - Oracle Machine Learning Notebooks
  - Oracle REST Data Services
  - Oracle Database API for MongoDB

  Additional configuration is required to access these Autonomous Database tools from on-premises environments. See Example: Connecting from Your Data Center to Autonomous Database to learn more.

  Accessing Oracle APEX, Database Actions, Oracle Graph Studio, or Oracle REST Data Services using a private endpoint from on-premises environments without completing the additional private endpoint configuration shows the error:

  ```
  404 Not Found
  ```
- After you update the network access to use a private endpoint, the URL for the Database Tools is different compared to using a public endpoint. You can find the updated URLs on the console, after changing from a public endpoint to a private endpoint.
- In addition to the default Oracle REST Data Services (ORDS) preconfigured with Autonomous Database, you can configure an alternative ORDS deployment that provides more configuration options and that can be used with private endpoints. See About Customer Managed Oracle REST Data Services on Autonomous Database to learn about an alternative ORDS deployment that can be used with private endpoints.
- Modifying a private IP address is not allowed after you provision or clone an instance, whether the IP address is automatically assigned when you enter a value in the **Private IP address** field.

Private Endpoints Configuration Examples on Autonomous Database

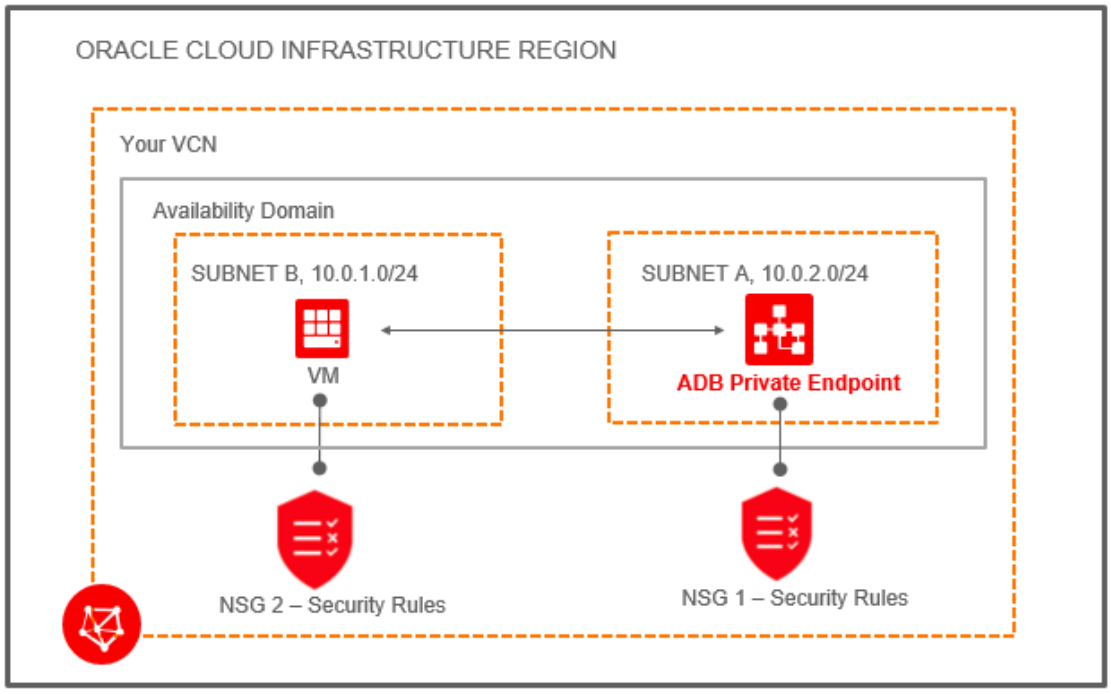
Shows several Private Endpoint (VCN) configuration samples for Autonomous Database.

- Example: Connecting from Inside Oracle Cloud Infrastructure VCN
  Demonstrates an application running inside Oracle Cloud Infrastructure on a virtual machine (VM) in the same VCN which is configured with your Autonomous Database.

- Example: Connecting from Your Data Center to Autonomous Database
  Demonstrates how to connect privately to an Autonomous Database from your on-premise data center. In this scenario, traffic never goes over the public internet.

Example: Connecting from Inside Oracle Cloud Infrastructure VCN

Demonstrates an application running inside Oracle Cloud Infrastructure on a virtual machine (VM) in the same VCN which is configured with your Autonomous Database.

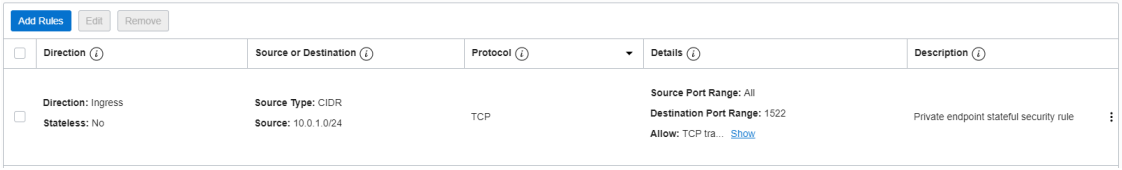


There is an Autonomous Database instance which has a private endpoint in the VCN named "Your VCN". The VCN includes two subnets: "SUBNET B" (CIDR 10.0.1.0/24) and "SUBNET A" (CIDR 10.0.2.0/24).

The Network Security Group (NSG) associated with the Autonomous Database instance is shown as "NSG 1 - Security Rules". This Network Security Group defines security rules that allow incoming and outgoing traffic to and from the Autonomous Database instance. Define a rule for the Autonomous Database instance as follows:

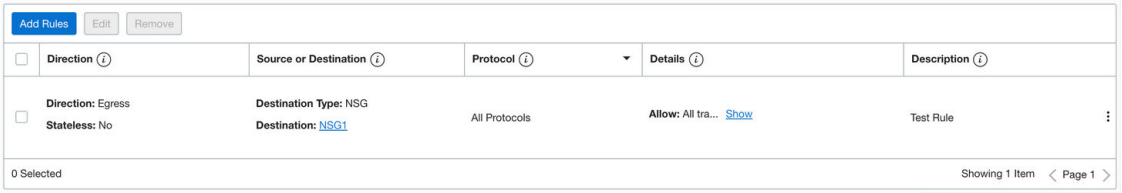
- For Mutual TLS authentication, add a stateful ingress rule to allow connections from the source to the Autonomous Database instance; the source is set to the address range you want to allow to connect to your database, IP Protocol is set to TCP, and the Destination Port Range is set to 1522.
- For TLS authentication, add a stateful ingress rule to allow connections from the source to the Autonomous Database instance; the source is set to the address range you want to allow to connect to your database, IP Protocol is set to TCP, and the Destination Port Range is set to 1521.
- To use Oracle APEX, Database Actions, and Oracle REST Data Services, add port 443 to the NSG rule.

The following figure shows a sample stateful security rule to control traffic for the Autonomous Database instance:



The application connecting to the Autonomous Database is running on a VM in SUBNET B. You also add a security rule to allow traffic to and from the VM (as shown, with label "NSG 2 Security Rules"). You can use a stateful security rule for the VM, so simply add a rule for egress to NSG 2 Security Rules (this allows access to the destination subnet A).

The following figure shows sample security rules that control traffic for the VM:

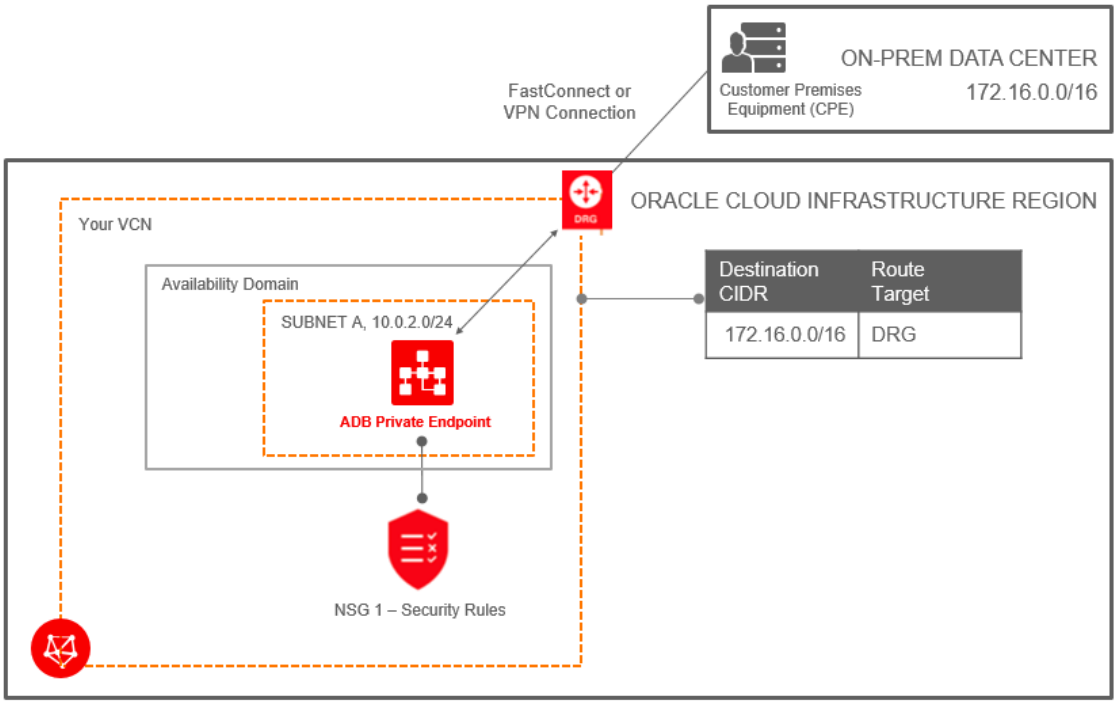


After you configure the security rules, your application can connect to the Autonomous Database instance using the client credentials wallet. See Download Client Credentials (Wallets) for more information.

See Network Security Groups for information on configuring Network Security Groups.

Example: Connecting from Your Data Center to Autonomous Database

Demonstrates how to connect privately to an Autonomous Database from your on-premise data center. In this scenario, traffic never goes over the public internet.



To connect from your data center, you connect the on-premise network to the VCN with FastConnect and then set up a Dynamic Routing Gateway (DRG). To resolve the Autonomous Database private endpoint, a Fully Qualified Domain Name (FQDN), requires that you add an

entry in your on-premise client's hosts file. For example, `/etc/hosts` file for Linux machines. For example:

```
/etc/hosts entry -> 10.0.2.7 example.adb.ca-toronto-1.oraclecloud.com
```

To use Oracle APEX, Database Actions, and Oracle REST Data Services, add another entry with the same IP. For example:

```
/etc/hosts entry -> 10.0.2.7 example.adb.ca-toronto-1.oraclecloudapps.com
```

You find the private endpoint IP and the FQDN as follows:

- The Private IP is shown on the Oracle Cloud Infrastructure console Autonomous Database details page for the instance.
- The FQDN is shown in the `tnsnames.ora` file in the Autonomous Database client credential wallet.

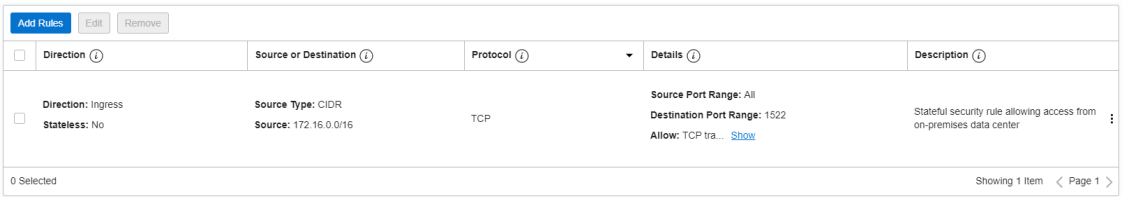
Alternatively you can use Oracle Cloud Infrastructure private DNS to provide DNS name resolution. See Private DNS for more information.

In this example there is a Dynamic Routing Gateway (DRG) between the on-premise data center and "Your VCN". The VCN contains the Autonomous Database. This also shows a route table for the VCN associated with the Autonomous Database, for outgoing traffic to CIDR 172.16.0.0/16 through the DRG.

In addition to setting up the DRG, define a Network Security Group (NSG) rule to allow traffic to and from the Autonomous Database, by adding a rule for the data center CIDR range (172.16.0.0/16). In this example, define a security rule in "NSG 1" as follows:

- For Mutual TLS authentication, create a stateful rule to allow ingress traffic from the data center. This is a stateful ingress rule with the source set to the address range you want to allow to connect to your database, protocol set to TCP, source port range set to CIDR range (172.16.0.0/16), and destination port set to 1522.
- For TLS authentication, create a stateful rule to allow ingress traffic from the data center. This is a stateful ingress rule with the source set to the address range you want to allow to connect to your database, protocol set to TCP, source port range set to CIDR range (172.16.0.0/16), and destination port set to 1521.
- To use Oracle APEX, Database Actions, and Oracle REST Data Services, add port 443 to the NSG rule.

The following figure shows the security rule that controls traffic for the Autonomous Database instance:



After you configure the security rule, your on-premise database application can connect to the Autonomous Database instance using the client credentials wallet. See Download Client Credentials (Wallets) for more information.

Update Network Options to Allow TLS or Require Only Mutual TLS (mTLS) Authentication on Autonomous Database

Describes how to update the secure client connection authentication options, Mutual TLS (mTLS) and TLS.

- Network Access Prerequisites for TLS Connections
  Describes the network access configuration prerequisites for TLS connections.
- Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication
- Update your Autonomous Database Instance to Require mTLS and Disallow TLS Authentication
  If your Autonomous Database instance is configured to allow TLS connections, you can update the instance to require mTLS connections and disallow TLS connections.

Network Access Prerequisites for TLS Connections

Describes the network access configuration prerequisites for TLS connections.

To allow an Autonomous Database instance to use TLS connections, either ACLs must be defined or a private endpoint must be configured:

- When an Autonomous Database instance is configured to operate over the public internet, one or more Access Control Lists (ACLs) must be defined before you use TLS authentication to connect to the database. To validate that ACLs are defined, in the **Network** area on the Autonomous Database Details page view the **Access Control List** field. This field shows **Enabled** when ACLs are defined and shows **Disabled** when ACLs are not defined.

  See Configuring Network Access with Access Control Rules (ACLs) for more information.
- When an Autonomous Database instance is configured with a private endpoint you can use TLS authentication to connect to the database. To validate that a private endpoint is defined, in the **Network** area on the Autonomous Database Details page view the **Access Type** field. This field shows **Virtual Cloud Network** when a private endpoint is defined.

  See Configure Network Access with Private Endpoints for more information.

> **Note:**
>
> When an Autonomous Database instance is configured with the network access type: **Allow secure access from everywhere**, you can only use TLS connections to connect to the database if you specify ACLs to restrict access.

Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication

If your Autonomous Database instance is configured to only allow mTLS connections, you can update the instance to allow both mTLS and TLS connections.

When you update your configuration to allow both mTLS and TLS, you can use both authentication types at the same time and connections are no longer restricted to require mTLS authentication.

You can allow TLS connections when network access is configured as follows:

- With network access configured with ACLs defined.

- With network access configured with a private endpoint defined.

> **Note:**
>
> When you configure your Autonomous Database instance network access with ACLs or a private endpoint, the ACLs or the private endpoint apply for both mTLS and TLS connections.

Perform the network access configuration prerequisites. See Network Access Prerequisites for TLS Connections for more information.

Perform the following steps as necessary:

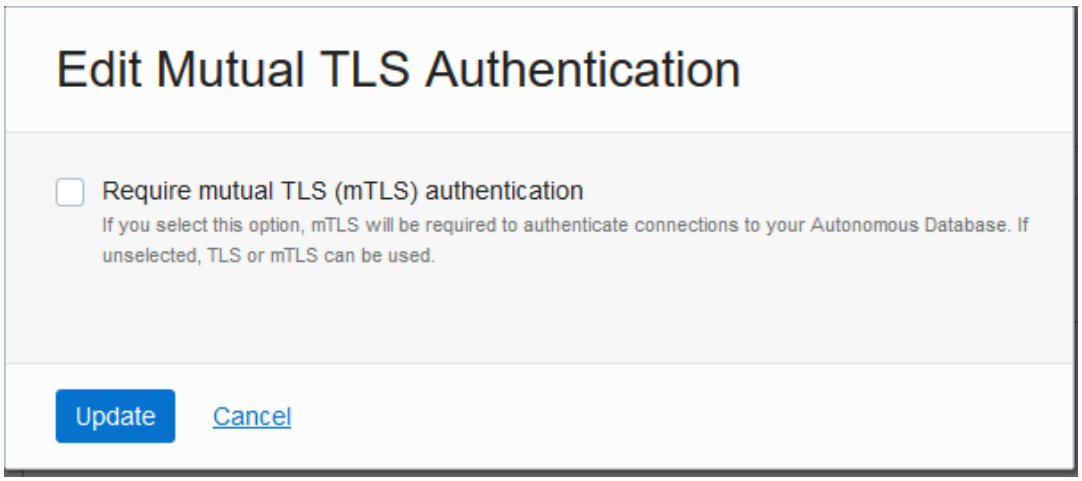
- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To change the Autonomous Database instance to allow TLS authentication, do the following:

1. On the **Autonomous Database Details** page, under **Network**, click **Edit** in the **Mutual TLS (mTLS) Authentication** field.

   This shows the Edit Mutual TLS Authentication page.

2. To change the value to allow TLS authentication, deselect **Require mutual TLS (mTLS) authentication**.



3. Click **Update**.

   The Autonomous Database Lifecycle State changes to **Updating**.

   After some time, the **Lifecycle State** shows **Available** and the **Mutual TLS (mTLS) Authentication** field changes to show **Not Required**.

After you define ACLs or configure a private endpoint and the **Mutual TLS (mTLS) Authentication** field shows **Not Required**, the ACLs or the private endpoint you specify apply to all connection types (mTLS and TLS).

Depending on the type of client, TLS connections have the following support with Autonomous Database:

- If the client is connecting with JDBC Thin using TLS authentication, the client can connect without providing a wallet. See Connect with JDBC Thin Driver for more information.
- If the client is connecting with managed ODP.NET or ODP.NET Core versions 19.13 or 21.4 (or above) using TLS authentication, the client can connect without providing a wallet. See Connect Microsoft .NET, Visual Studio Code, and Visual Studio Without a Wallet for more information.
- If the client is connecting with SQLNet and Oracle Call Interface (OCI), and for certain other connection types with TLS authentication, the clients must provide the CA certificate in a wallet. See Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections Using TLS Authentication and Connect SQL*Plus Without a Wallet for more information.

Update your Autonomous Database Instance to Require mTLS and Disallow TLS Authentication

If your Autonomous Database instance is configured to allow TLS connections, you can update the instance to require mTLS connections and disallow TLS connections.

> **Note:**
>
> When you update an Autonomous Database instance to require Mutual TLS (mTLS) connections, existing TLS connections are disconnected.

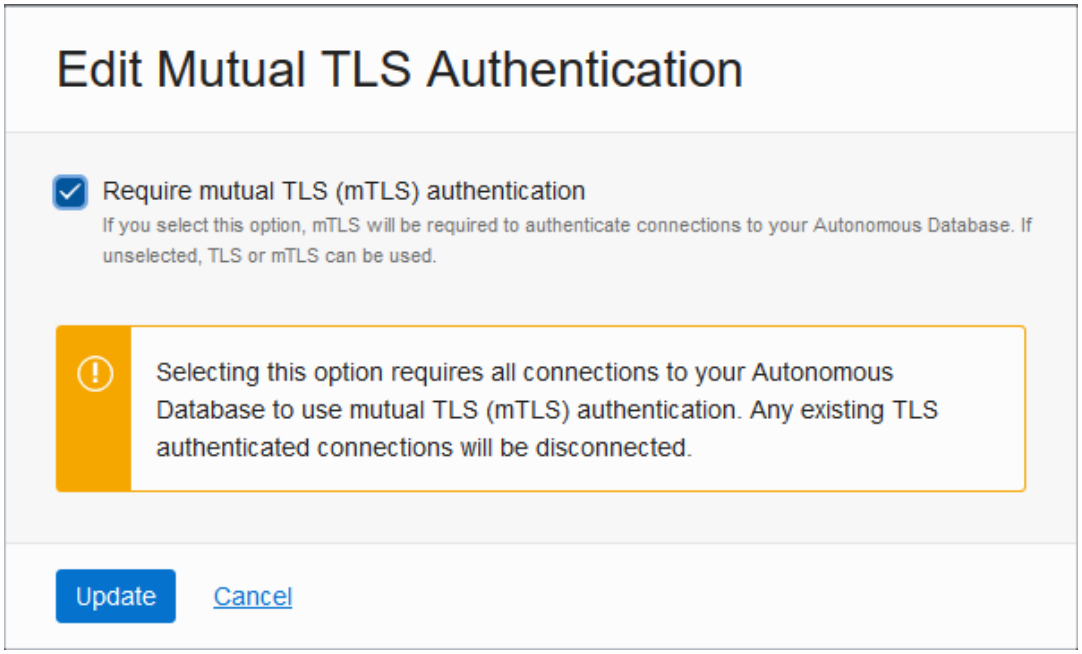
Perform the following steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display name** column.

To change the Autonomous Database instance to require mTLS authentication and to not allow TLS authentication, do the following:

1. On the **Autonomous Database Details** page, under **Network**, click **Edit** in the **Mutual TLS (mTLS) Authentication** field.

   This shows the Edit Mutual TLS Authentication page.
2. Select **Require mutual TLS (mTLS) authentication**.



3. Click **Update**.

   The Autonomous Database Lifecycle State changes to **Updating**.

   After some time, the **Lifecycle State** shows **Available** and the **Mutual TLS (mTLS) Authentication** field changes to show **Required**.

Audit Autonomous Database

Autonomous Database provides auditing that allows you to monitor Oracle database activities.

- About Auditing Autonomous Database
  Autonomous Database provides auditing to track, monitor, and record database actions. Auditing can help you detect security risks and improve regulatory compliance for your database.
- Register Oracle Data Safe on Autonomous Database
  Use Oracle Data Safe to apply auditing policies for database users, for administrative users, to apply predefined auditing policies or to extend the audit data record retention for your Autonomous Database instance.
- Extend Audit Record Retention with Oracle Data Safe on Autonomous Database
  Use Oracle Data Safe to extend the audit data record retention to a specified number of months.
- View and Manage Oracle Data Safe Audit Trails on Autonomous Database
  Data Safe uses audit trails to define where to retrieve the audit data and to collect Autonomous Database audit records. During the registration process, Oracle Data Safe discovers the audit trails and creates an audit trail resource.
- View and Manage Audit Policies with Oracle Data Safe on Autonomous Database
  Use Oracle Data Safe to set audit policies for your Autonomous Database instance.
- Generate Audit Reports with Data Safe on Autonomous Database
  Data Safe includes out-of-box audit data reports, and you can create custom reports to suit your needs.

About Auditing Autonomous Database

Autonomous Database provides auditing to track, monitor, and record database actions. Auditing can help you detect security risks and improve regulatory compliance for your database.

- Audit Features on Autonomous Database
  Autonomous Database includes extensive, sophisticated audit capabilities that allow you capture the audit information you need for your organization. Autonomous Database provides default auditing.
- Audit Data on Autonomous Database
  Autonomous Database protects audit data and writes its audit trail to the `UNIFIED_AUDIT_TRAIL` data dictionary view.
- Default Audit Policies on Autonomous Database
  Autonomous Database provides auditing to track, monitor, and record activities on your database.

Audit Features on Autonomous Database

Autonomous Database includes extensive, sophisticated audit capabilities that allow you capture the audit information you need for your organization. Autonomous Database provides default auditing.

In addition, you can use either of the following to apply auditing policies:

- Use Oracle Data Safe to apply auditing policies for database users, for administrative users, and to apply predefined auditing policies or to apply customized auditing policies. See Activity Auditing Overview for more information.
- Configure Oracle Database Audit Policies. See Configuring Audit Policies for more information.

You can configure auditing to accomplish the following:

- Enable accountability for actions. These include actions taken in a particular schema, table, or row, or affecting specific content.
- Deter users, or others, such as intruders, from inappropriate actions based on their accountability.
- Investigate suspicious activity. For example, if a user is logging into the database using the application's database credentials, then auditing connections to the database lets you determine that the login came from a user's workstation instead of from the application server.
- Notify an auditor of the actions of an unauthorized user. For example, notify an auditor when an unauthorized user attempts to delete data from a table.
- Monitor and gather data about specific database activities. For example, you can gather statistics about which tables are being updated, the number of failed logins, or how many concurrent users connect at peak times.
- Detect problems with an authorization or access control implementation. For example, you can create audit policies that you expect will never generate an audit record because the data is protected in other ways. However, if these policies generate audit records, then you will know the other security controls are not properly implemented.
- Address auditing requirements for compliance. Regulations such as the following have common auditing-related requirements:

- European Union General Data Protection Regulation (GDPR)
- Sarbanes-Oxley Act
- Health Insurance Portability and Accountability Act (HIPAA)
- International Convergence of Capital Measurement and Capital Standards: a Revised Framework (Basel II)
- Japan Privacy Law
- European Union Directive on Privacy and Electronic Communications

Audit Data on Autonomous Database

Autonomous Database protects audit data and writes its audit trail to the `UNIFIED_AUDIT_TRAIL` data dictionary view.

The underlying table storing audit data on Autonomous Database is `AUDSYS.AUD$UNIFIED`. This table is protected and does not allow users to perform DML/DDL operations or to purge the table (any attempt to perform these actions automatically produces an audit record). After an audit record is written, the only activity allowed is for the ADMIN user to perform a `PURGE`. The ADMIN has the AUDIT_ADMIN role that is required to run a `PURGE`. If you assign the AUDIT_ADMIN role to another user, then that user could also perform a `PURGE`.

Depending on the number and type of audit policies you use and the amount of activity, over time the audit trail can grow to use a large amount of storage. Autonomous Database provides the following ways to limit the storage required for audit data:

- Each Autonomous Database instance runs an automated purge job once a day to remove all audit records older than fourteen (14) days.
- Users with the AUDIT_ADMIN role can purge audit records manually using the `DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL` procedure. See DBMS_AUDIT_MGMT for more information.

If you need a longer audit data retention period than 14 days, use Oracle Data Safe to retain audit data. See Extend Audit Record Retention with Oracle Data Safe on Autonomous Database for more information.

Autonomous Database audits and logs every operation carried out in your database by the Oracle Cloud Infrastructure Operations teams. See View Oracle Cloud Infrastructure Operations Actions for more information on how to audit Operations activities.

Default Audit Policies on Autonomous Database

Autonomous Database provides auditing to track, monitor, and record activities on your database.

By default, Autonomous Database applies audit policies to audit the following database activities:

- All activity by Oracle Cloud Operations
- All login failures to the database
- All password changes
- Attempts to create or alter procedures
- Execution of certain procedures, including procedures in the packages: `UTL_HTTP` or `UTL_SMTP` that connect to the network

In addition, you can use either of the following to apply additional auditing policies:

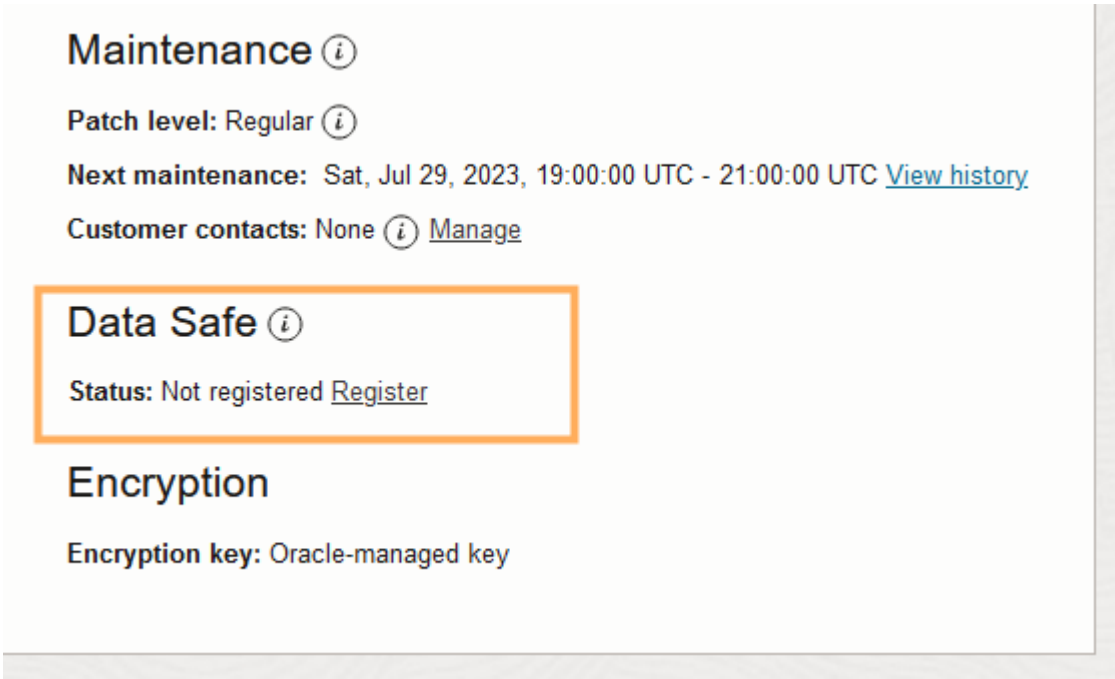
- Use Oracle Data Safe to apply auditing policies for database users, for administrative users, and to apply predefined auditing policies or to apply customized auditing policies. For more information, see:
  - View and Manage Audit Trails
  - View and Manage Audit Policies
- Configure Oracle Database Audit Policies. See Configuring Audit Policies for more information.

Register Oracle Data Safe on Autonomous Database

Use Oracle Data Safe to apply auditing policies for database users, for administrative users, to apply predefined auditing policies or to extend the audit data record retention for your Autonomous Database instance.

Register your Autonomous Database instance with Oracle Data Safe as follows:

1. Access your Autonomous Database instance from the Oracle Cloud Infrastructure Console.
   a. Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
   b. From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
   c. On the Autonomous Databases page select an Autonomous Database from the links under the **Display Name** column.
2. Register your Autonomous Database instance with Oracle Data Safe.
   a. On the Autonomous Database Details page, under **Data Safe**, click **Register**.



b. In the Register database with Data Safe dialog, click **Confirm**.

The Data Safe status shows: **Registering**. This step takes about 15 to 20 minutes.

After Oracle Data Safe is registered, the Data Safe status shows **Registered** and two links: **View** and **Deregister**.

Click **View** to show the Data Safe register database details page.

Click **Deregister** to disable Oracle Data Safe.

Extend Audit Record Retention with Oracle Data Safe on Autonomous Database

Use Oracle Data Safe to extend the audit data record retention to a specified number of months.

First register your Autonomous Database instance with Oracle Data Safe. See Register Oracle Data Safe on Autonomous Database for more information.

After your Autonomous Database instance is registered, you can specify the Data Safe retention period.

See Update Retention Periods for a Target Database for more information.

View and Manage Oracle Data Safe Audit Trails on Autonomous Database

Data Safe uses audit trails to define where to retrieve the audit data and to collect Autonomous Database audit records. During the registration process, Oracle Data Safe discovers the audit trails and creates an audit trail resource.

Oracle Data Safe lists resources on the Audit Trails page. To access the Audit Trails page, under Security Center in Data Safe click **Activity Auditing**, and then, on the Activity Auditing page under **Related Resources** click **Audit Trails**. You can discover new audit trails at any time and remove audit trail resources in Oracle Data Safe as needed.

First register your Autonomous Database instance with Oracle Data Safe. See Register Oracle Data Safe on Autonomous Database for more information.

When the Autonomous Database is stopped or restarted, the following happens:

- The audit trail switches to a retrying state and Data Safe makes multiple attempts to reconnect for four (4) hours. The Audit Trail **Collection State** field shows: **RETRYING**.

  In this case, if the Autonomous Database (target database) starts, the audit trail automatically resumes.

- After four hours, the audit trail switches to a stopped state. The Audit Trail **Collection State** field shows: **STOPPED_NEEDS_ATTN**.

  In this case, when the Autonomous Database (target database) starts and the audit trail collection state is **STOPPED_NEEDS_ATTN**, you can manually resume the audit trail.

You can also manually stop or delete the audit trail. Deleting the audit trail does not remove audit records that have already been collected. Those records remain in Data Safe until the retention period is reached.

See View and Manage Audit Trails for more information.

View and Manage Audit Policies with Oracle Data Safe on Autonomous Database

Use Oracle Data Safe to set audit policies for your Autonomous Database instance.

First register your Autonomous Database instance with Oracle Data Safe. See Register Oracle Data Safe on Autonomous Database for more information.

After your Autonomous Database instance is registered, access Oracle Data Safe to set audit policies.

See View and Manage Audit Policies for more information.

Generate Audit Reports with Data Safe on Autonomous Database

Data Safe includes out-of-box audit data reports, and you can create custom reports to suit your needs.

After you enable and register Oracle Data Safe, and you add a trail to collect audit data from your Autonomous Database instance, then you can use the reports to monitor activity for your database.

See View and Manage Audit Reports for more information.

Rotate Wallets for Autonomous Database

Wallet rotation lets you invalidate existing client certification keys for a database instance or for all Autonomous Database instances that a cloud account owns in a region.

- About Wallet Rotation
  You have the option to perform one of two types of wallet rotation: immediate or with a grace period.
- Rotate Wallets with Immediate Rotation
  Immediate wallet rotation lets you invalidate existing client certification keys for an Autonomous Database instance or for all Autonomous Database instances that a cloud account owns in a region.
- Rotate Wallets with Grace Period

About Wallet Rotation

You have the option to perform one of two types of wallet rotation: immediate or with a grace period.

- **Immediate** wallet rotation initiates immediately, without delay.
- **After a grace period** wallet rotation occurs with a grace period. During the grace period the old client certification keys remain valid for a selected time of 1 hour to 24 hours. After the grace period expires only the new client certification keys are valid.

You may want to rotate wallets for any of the following reasons:

- If your organization's policies require regular client certification key rotation.
- When a client certification key or a set of keys is suspected to be compromised.

Rotate Wallets with Immediate Rotation

Immediate wallet rotation lets you invalidate existing client certification keys for an Autonomous Database instance or for all Autonomous Database instances that a cloud account owns in a region.

There are two options for immediate client certification key rotation:

- Per-database with **Instance wallet** selected:
  - For the database whose certification key is rotated, any existing database specific instance wallets will be void. After you rotate a wallet you have to download a new wallet to connect to the database.
  - Regional wallets containing all database certification keys continue to work.
  - All user sessions are terminated for the database whose wallet is rotated. User session termination begins after wallet rotation completes, however this process does not happen immediately.

> **Note:**
>
> If you want to terminate all connections immediately after the wallet rotation completes, Oracle recommends that you restart the Autonomous Database instance. This provides the highest level of security for your database.

- Regional level with **Regional wallet** selected:
  - For the region whose certification key is rotated, both regional and database specific instance wallets will be void. After you rotate a wallet you have to download new regional or instance wallets to connect to any database in the region.
  - All user sessions are terminated for the databases in the region whose wallet is rotated. User session termination begins after wallet rotation completes, however this process does not happen immediately.

> **Note:**
>
> If you want to terminate all connections immediately after the wallet rotation completes, Oracle recommends that you restart the Autonomous Database instances in the region. This provides the highest level of security for your database.

To immediately rotate the client certification key for a given database or for all Autonomous Database instances that a cloud account owns in a region:

1. Navigate to the Autonomous Database details page.
2. Click **Database connection**.
3. On the **Database connection** page select the **Wallet type**:
   - **Instance wallet**: Wallet rotation for a single database only; this provides a database-specific wallet rotation.
   - **Regional wallet**: Wallet rotation for all Autonomous Databases for a given tenant and region (this option rotates the client certification key for all service instances that a cloud account owns).

4. Click **Rotate wallet**.
5. Enter the name as shown in the dialog to confirm the wallet rotation.
6. In the Rotate Wallet dialog, click **Rotate**.

The Database Connection page shows: **Rotation in Progress**.

After the rotation completes, the **Wallet last rotated** field shows the last rotation date and time.

Oracle recommends you provide a database-specific instance wallet to end users and for application use whenever possible, with Wallet type set to **Instance wallet** when you use **Download wallet**. Regional wallets should only be used for administrative purposes that require potential access to all Autonomous Databases within a region.

You can also use the Autonomous Database API to rotate wallets using `UpdateAutonomousDatabaseRegionalWallet` and `UpdateAutonomousDatabaseWallet`. See Autonomous Database Wallet Reference for more information.

Rotate Wallets with Grace Period

Autonomous Database allows you to rotate wallets for an Autonomous Database instance or for all instances that a cloud account owns in a region. with a grace period of 1 hour to 24 hours.

Setting a grace period allows you to perform wallet rotation without down time. During the grace period you can inform users to download the new wallet and to update their applications to use the new wallet. During the grace period both the old and new client certification keys are valid. When the grace period expires, Autonomous Database invalidates the old client certification keys and only the new client certification keys are valid.

There are two options for client certification key rotation with a grace period:

- Per-database with **Instance wallet** selected:
  - For the database whose certification key is rotated, database specific instance wallets that were in use before the wallet rotation be void after the grace period expires.
  - After you perform client certification key rotation with a grace period, you can immediately download a wallet and use the new wallet to connect to the database.
  - Regional wallets containing all database certification keys continue to work.
  - After the grace period expires, existing connections using the old wallet continue to work.

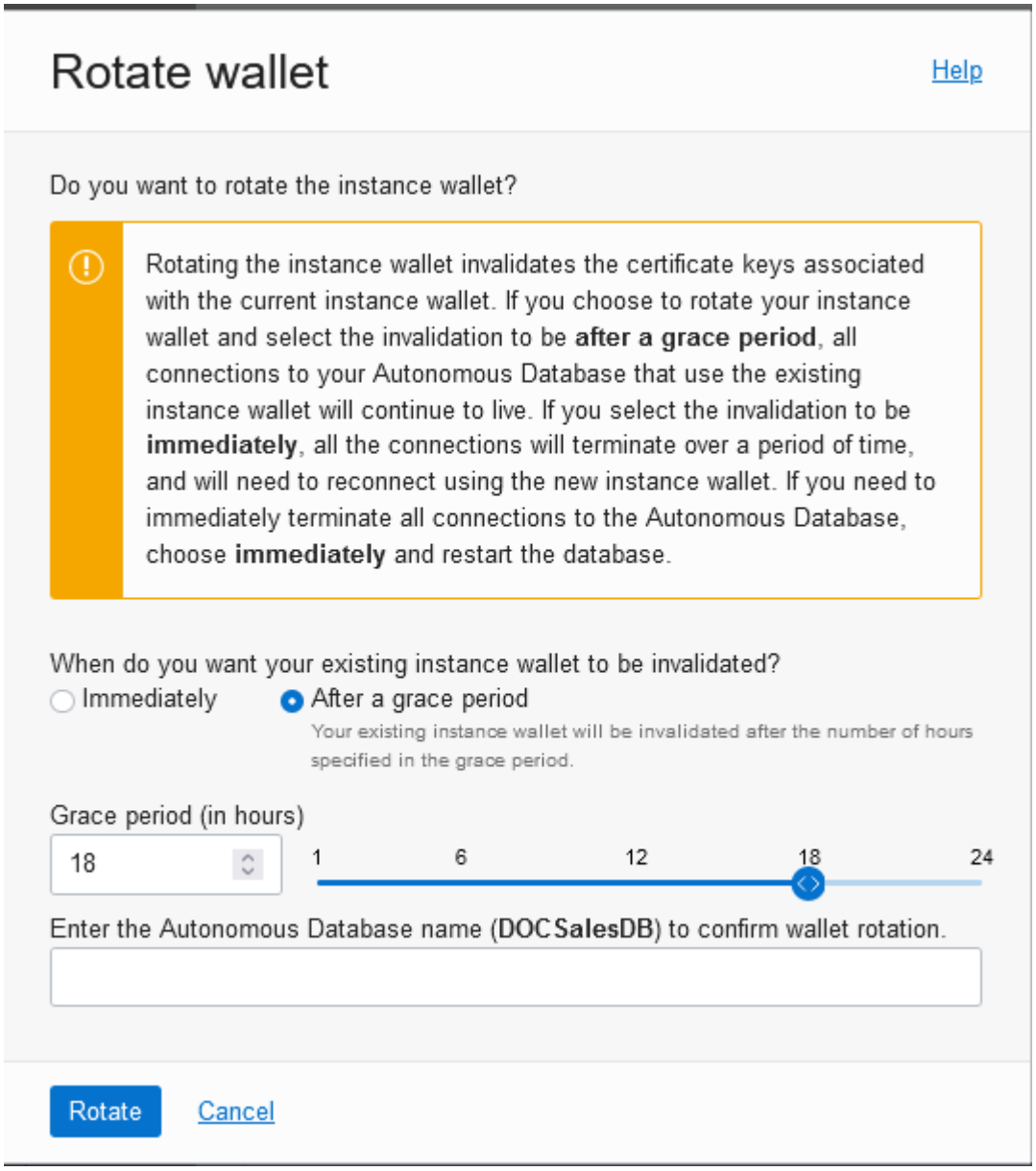
> **Note:**
>
> After the grace period completes, if you want to terminate any connections using the old wallet, Oracle recommends that you restart the Autonomous Database instance.

- Regional level with **Regional wallet** selected:
  - For the region whose certification key is rotated, both regional and database specific instance wallets will be void. After the grace period expires you have to download new regional or instance wallets to connect to any database in the region.
  - After the grace period expires, existing connections using the old wallets continue to work.

> **Note:**
>
> After the grace period completes, if you want to terminate any connections using the old wallets, Oracle recommends that you restart every Autonomous Database instance in the region.

To rotate the client certification key with a grace period for a given database or for all for all Autonomous Database instances that a cloud account owns in a region:

1. Navigate to the Autonomous Database details page.
2. Click **Database connection**.
3. On the **Database connection** page select the **Wallet type**:
   - **Instance wallet**: Wallet rotation for a single database only; this provides a database-specific wallet rotation.
   - **Regional wallet**: Wallet rotation for all Autonomous Databases for a given tenant and region (this option rotates the client certification key for all service instances in the region that a cloud account owns).
4. Click **Rotate wallet**.
5. Select **After a grace period**.
6. In the **Grace period (in hours)** area, either enter a value in the text field or use the slider to select a value.



7. Enter the name as shown in the dialog to confirm the wallet rotation.
8. In the Rotate Wallet dialog, click **Rotate**.

The Database Connection page shows: **Rotation in Progress**.

After the rotation completes, the **Wallet last rotated** field shows the last rotation date and time.

Notes for wallet rotation with a grace period:

- Always Free Autonomous Databases only support immediate wallet rotation (wallet rotation with a grace period is not supported).
- Oracle recommends you provide a database-specific instance wallet to end users and for application use whenever possible, with Wallet type set to **Instance wallet** when you use **Download wallet**. Regional wallets should only be used for administrative purposes that require potential access to all Autonomous Databases within a region.

You can also use the Autonomous Database API to rotate wallets using `UpdateAutonomousDatabaseRegionalWallet` and `UpdateAutonomousDatabaseWallet`. See Autonomous Database Wallet Reference for more information.

Use Oracle Database Vault with Autonomous Database

Oracle Database Vault implements powerful security controls for your database. These unique security controls restrict access to application data by privileged database users, reducing the risk of insider and outside threats and addressing common compliance requirements.

See What Is Oracle Database Vault? for more information.

- Oracle Database Vault Users and Roles on Autonomous Database
  Oracle Database Vault provides powerful security controls to help protect application data from unauthorized access, and implement separation of duties between administrators and data owners to comply with privacy and regulatory requirements.
- Enable Oracle Database Vault on Autonomous Database
  Shows the steps to enable Oracle Database Vault on Autonomous Database.
- Disable Oracle Database Vault on Autonomous Database
  Shows the steps to disable Oracle Database Vault on Autonomous Database.
- Disable User Management with Oracle Database Vault on Autonomous Database
  Shows how to disallow user management related operations for specified components on Autonomous Database with Oracle Database Vault enabled.
- Enable User Management with Oracle Database Vault on Autonomous Database
  Shows the steps to allow user management for a specified component on Autonomous Database with Oracle Database Vault enabled.

Oracle Database Vault Users and Roles on Autonomous Database

Oracle Database Vault provides powerful security controls to help protect application data from unauthorized access, and implement separation of duties between administrators and data owners to comply with privacy and regulatory requirements.

By default the ADMIN user has the `DV_OWNER` and `DV_ACCTMGR` roles. If you want to set up separate users for `DV_OWNER` and `DV_ACCTMGR` accounts, see Oracle Database Vault Schemas, Roles, and Accounts.

The user management is by default enabled for the APEX component when Oracle Database Vault is enabled. When user management is enabled, the APEX users who have the necessary roles to `CREATE | ALTER | DROP` users have the needed privileges to perform these operations when Database Vault is enabled. To change this, see Disable User Management with Oracle Database Vault on Autonomous Database.

On Autonomous Database with Oracle Database Vault enabled, grant the following privileges:

- When using Oracle GoldenGate, grant the `GGADMIN` user `DV_GOLDENGATE_ADMIN` and `DV_GOLDENGATE_REDO_ACCESS`.
- The ADMIN user must grant the `BECOME USER` privilege to users who need to use Oracle Data Pump. To perform some Oracle Data Pump operations additional Oracle Database Vault authorization may be needed. For example to run a full database export or to export a realm protected schema requires using `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER`.

  See AUTHORIZE_DATAPUMP_USER Procedure for more information.

Enable Oracle Database Vault on Autonomous Database

Shows the steps to enable Oracle Database Vault on Autonomous Database.

Oracle Database Vault is disabled by default on Autonomous Database. To configure and enable Oracle Database Vault on Autonomous Database, do the following:

1. Configure Oracle Database Vault using the following command:

   ```
   EXEC DBMS_CLOUD_MACADM.CONFIGURE_DATABASE_VAULT('adb_dbv_owner',
   'adb_dbv_acctmgr');
   ```

   Where:

   - *adb_dbv_owner* is the Oracle Database Vault owner.
   - *adb_dbv_acctmgr* is the account manager.

   See CONFIGURE_DATABASE_VAULT Procedure for more information.
2. Enable Oracle Database Vault:

   ```
   EXEC DBMS_CLOUD_MACADM.ENABLE_DATABASE_VAULT;
   ```

   See ENABLE_DATABASE_VAULT Procedure for more information.
3. Restart the Autonomous Database instance.

   See Restart Autonomous Database for more information.

Use the following command to check if Oracle Database Vault is enabled or disabled:

```
SELECT * FROM DBA_DV_STATUS;
```

Output similar to the following appears:

```
NAME                 STATUS
-------------------- -----------
DV_CONFIGURE_STATUS  TRUE
DV_ENABLE_STATUS     TRUE
```

The `DV_ENABLE_STATUS` value `TRUE` indicates Oracle Database Vault is enabled.

> **Note:**
>
> Autonomous Database maintenance operations such as backups and patching are not affected when Oracle Database Vault is enabled.

See Disable Oracle Database Vault on Autonomous Database for information on disabling Oracle Database Vault.

Disable Oracle Database Vault on Autonomous Database

Shows the steps to disable Oracle Database Vault on Autonomous Database.

To disable Oracle Database Vault on Autonomous Database, do the following:

1. Disable Oracle Database Vault.

   ```
   EXEC DBMS_CLOUD_MACADM.DISABLE_DATABASE_VAULT;
   ```

   See DISABLE_DATABASE_VAULT Procedure for more information.
2. Restart the Autonomous Database instance.

   See Restart Autonomous Database for more information.

Use the following command to check if Oracle Database Vault is enabled or disabled:

```
SELECT * FROM DBA_DV_STATUS;
```

Output similar to the following appears:

```
NAME                 STATUS
-------------------- -----------
DV_CONFIGURE_STATUS  TRUE
DV_ENABLE_STATUS     FALSE
```

The `DV_ENABLE_STATUS` value `FALSE` indicates Oracle Database Vault is enabled.

Disable User Management with Oracle Database Vault on Autonomous Database

Shows how to disallow user management related operations for specified components on Autonomous Database with Oracle Database Vault enabled.

Autonomous Database with Oracle Database Vault enabled has user management, by default, enabled for the Oracle APEX console. If you want to enforce stricter separation of duty and disallow user management from this console, use `DBMS_CLOUD_MACADM.DISABLE_USERMGMT_DATABASE_VAULT`.

1. As a user granted `DV_ACCTMGR` and `DV_ADMIN` roles you can disable user management for specified components.
2. To disable user management for a specified component, for example for the APEX component, use the following command:

   ```
   EXEC DBMS_CLOUD_MACADM.DISABLE_USERMGMT_DATABASE_VAULT('APEX');
   ```

See DISABLE_USERMGMT_DATABASE_VAULT Procedure for more information.

Enable User Management with Oracle Database Vault on Autonomous Database

Shows the steps to allow user management for a specified component on Autonomous Database with Oracle Database Vault enabled.

Autonomous Database with Oracle Database Vault enabled has user management, by default, enabled for the Oracle APEX console. This allows user management for operations such as

`CREATE USER`, `ALTER USER`, and `DROP USER` from the specified component in Autonomous Database.

Use `DBMS_CLOUD_MACADM.ENABLE_USERMGMT_DATABASE_VAULT` to allow specified user accounts to perform user management when Oracle Database Vault is enabled. Use this procedure if user management is disabled and you want to enable it again.

1. A user granted `DV_ACCTMGR` and `DV_ADMIN` roles can enable user management for specified components.
2. To enable user management for a specified component, for example for the APEX component, use the following command:

```
EXEC DBMS_CLOUD_MACADM.ENABLE_USERMGMT_DATABASE_VAULT('APEX');
```

See ENABLE_USERMGMT_DATABASE_VAULT Procedure for more information.

IAM Policies for Autonomous Database

Provides information on IAM policies required for API operations on Autonomous Database.

Oracle Autonomous Database relies on the IAM (Identity and Access Management) service to authenticate and authorize cloud users to perform operations that use any of the Oracle Cloud Infrastructure interfaces (the console, REST API, CLI, or SDK).

The IAM service uses **groups**, **compartments,** and **policies** to control which cloud users can access which resources.

- IAM Permissions and API Operations for Autonomous Database
  This topic covers the available IAM permissions for operations on Autonomous Database.
- Policy Details for Autonomous Database
  This topic covers details for writing policies to control access to Autonomous Database resources.
- Policies to Manage Autonomous Databases
  Provides a list of the IAM policies required for a cloud user to perform management operations on Autonomous Databases.

IAM Permissions and API Operations for Autonomous Database

This topic covers the available IAM permissions for operations on Autonomous Database.

The following are the IAM permissions for Autonomous Database:

- `AUTONOMOUS_DATABASE_CONTENT_READ`
- `AUTONOMOUS_DATABASE_CONTENT_WRITE`
- `AUTONOMOUS_DATABASE_CREATE`

  See Cloning Permissions for additional cloning limitations.
- `AUTONOMOUS_DATABASE_DELETE`
- `AUTONOMOUS_DATABASE_INSPECT`
- `AUTONOMOUS_DATABASE_UPDATE`
- `AUTONOMOUS_DB_BACKUP_CONTENT_READ`
- `AUTONOMOUS_DB_BACKUP_CREATE`

- AUTONOMOUS_DB_BACKUP_INSPECT
- NETWORK_SECURITY_GROUP_UPDATE_MEMBERS
- VNIC_ASSOCIATE_NETWORK_SECURITY_GROUP

API Operation and Authorization Verb	Permissions Required to Use the Operation
AutonomousDatabaseManualRefresh manualRefreshAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
CancelAutonomousDatabaseSession cancelAutonomousDatabaseSession	AUTONOMOUS_DATABASE_CONTENT_WRITE
ChangeAutonomousDatabaseCompartment changeAutonomousDatabaseCompartment	Required on the source and the target compartment: AUTONOMOUS_DATABASE_UPDATE AUTONOMOUS_DB_BACKUP_CONTENT_READ AUTONOMOUS_DB_BACKUP_INSPECT AUTONOMOUS_DB_BACKUP_CREATE AUTONOMOUS_DATABASE_CONTENT_WRITE Required in both the source and the target compartment when Private Endpoint is enabled: VNIC_ASSOCIATE_NETWORK_SECURITY_GROUP NETWORK_SECURITY_GROUP_UPDATE_MEMBERS
ChangeDisasterRecoveryConfiguration changeDisasterRecoveryConfiguration	AUTONOMOUS_DATABASE_UPDATE
ConfigureAutonomousDatabaseVaultKey configureAutonomousDatabaseVaultKey	AUTONOMOUS_DATABASE_UPDATE
CreateAutonomousDatabaseBackup createAutonomousDatabaseBackup	AUTONOMOUS_DB_BACKUP_CREATE AUTONOMOUS_DATABASE_CONTENT_READ
CreateAutonomousDatabase createAutonomousDatabase	AUTONOMOUS_DATABASE_CREATE
DeleteAutonomousDatabaseBackup deleteAutonomousDatabaseBackup	AUTONOMOUS_DB_BACKUP_INSPECT AUTONOMOUS_DB_BACKUP_DELETE
DeleteAutonomousDatabase deleteAutonomousDatabase	AUTONOMOUS_DATABASE_DELETE
DeregisterAutonomousDatabaseDataSafe updateAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
DisableAutonomousDatabaseOperationsInsights updateAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
DisableDatabaseManagement updateAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
EnableAutonomousDatabaseOperationsInsights updateAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
EnableDatabaseManagement updateAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE

API Operation and Authorization Verb	Permissions Required to Use the Operation
FailOverAutonomousDatabase failOverAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
GenerateAutonomousDatabasePerformanceData generateAutonomousDatabasePerformanceData	AUTONOMOUS_DATABASE_CONTENT_READ
GenerateAutonomousDatabaseWallet generateAutonomousDatabaseWallet	AUTONOMOUS_DATABASE_CONTENT_READ
GetAutonomousDatabaseBackupConfig getAutonomousDatabaseBackupConfig	AUTONOMOUS_DATABASE_INSPECT
GetAutonomousDatabaseBackup getAutonomousDatabaseBackup	AUTONOMOUS_DB_BACKUP_INSPECT
GetAutonomousDatabaseCapability getAutonomousDatabaseCapabilities	AUTONOMOUS_DATABASE_INSPECT
GetAutonomousDatabaseConsoleToken getAutonomousDatabaseConsoleToken	AUTONOMOUS_DATABASE_CONTENT_WRITE
GetAutonomousDatabase getAutonomousDatabase	AUTONOMOUS_DATABASE_INSPECT
GetAutonomousDatabaseRegionalWallet getAutonomousDatabaseRegionalWallet	AUTONOMOUS_DATABASE_CONTENT_READ
GetAutonomousDatabaseWallet getAutonomousDatabaseWallet	AUTONOMOUS_DATABASE_CONTENT_READ
GetKeyDetail getDatabaseKeyDetails	N/A
ListAutonomousDatabaseBackups listAutonomousDatabaseBackups	AUTONOMOUS_DB_BACKUP_INSPECT
ListAutonomousDatabaseClones listAutonomousDatabaseClones	AUTONOMOUS_DATABASE_INSPECT
ListAutonomousDatabaseRefreshableClones ListAutonomousDatabaseRefreshableClones	AUTONOMOUS_DATABASE_INSPECT
ListAutonomousDatabases ListAutonomousDatabases	AUTONOMOUS_DATABASE_INSPECT
RegisterAutonomousDatabaseDataSafe updateAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
RestartAutonomousDatabase restartAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
RestoreAutonomousDatabase restoreAutonomousDatabase	AUTONOMOUS_DB_BACKUP_INSPECT AUTONOMOUS_DB_BACKUP_CONTENT_READ AUTONOMOUS_DATABASE_CONTENT_WRITE

API Operation and Authorization Verb	Permissions Required to Use the Operation
RetrieveDatabasePerformanceBulkData retrieveAutonomousDatabasePerformanceBulkData	AUTONOMOUS_DATABASE_CONTENT_READ
RotateAutonomousDatabaseEncryptionKey rotateDatabaseEncryptionKey	AUTONOMOUS_DATABASE_UPDATE
ShrinkAutonomousDatabase shrinkAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
StartAutonomousDatabase startAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
StopAutonomousDatabase stopAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
SwitchOverAutonomousDatabase switchoverAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
UpdateAutonomousDatabaseBackup updateAutonomousDatabaseBackup	AUTONOMOUS_DB_BACKUP_UPDATE
UpdateAutonomousDatabaseRegionalWallet updateAutonomousDatabaseRegionalWallet	AUTONOMOUS_DATABASE_UPDATE
UpdateAutonomousDatabase updateAutonomousDatabase	Three possible cases: • If Workload is NULL: AUTONOMOUS_DATABASE_UPDATE • If Workload is not NULL: AUTONOMOUS_DATABASE_CREATE AUTONOMOUS_DATABASE_UPDATE • If Tagging is enabled: AUTONOMOUS_DATABASE_UPDATE AUTONOMOUS_DATABASE_INSPECT
UpdateAutonomousDatabaseWallet updateAutonomousDatabaseWallet	AUTONOMOUS_DATABASE_UPDATE

Cloning Permissions

General IAM permissions are supported for Autonomous Database. In addition you can use `target.autonomous-database.cloneType` with the supported permission values to control the level of access, as shown in the following table.

target.autonomous-database.cloneType Value	Description
CLONE-FULL	Allow full clone only.
CLONE-METADATA	Allow metadata clone only.
CLONE-REFRESHABLE	Allow refreshable clone only.
/CLONE*/	Allow any kind of clone.

Example policies with the supported `target.autonomous-database.cloneType` permission values:

```
Allow group group-name to manage autonomous-databases in compartment id
compartment-ocid
    where all {request.permission =  'AUTONOMOUS_DATABASE_CREATE',
target.autonomous-database.cloneType = 'CLONE-FULL'}
```

```
Allow group group-name to manage autonomous-databases in compartment id
compartment-ocid
    where all {request.permission =  'AUTONOMOUS_DATABASE_CREATE',
target.autonomous-database.cloneType = 'CLONE-METADATA'}
```

```
Allow group group-name to manage autonomous-databases in compartment id
compartment-ocid
    where all {request.permission =  'AUTONOMOUS_DATABASE_CREATE',
target.autonomous-database.cloneType = 'CLONE-REFRESHABLE'}
```

```
Allow group group-name to manage autonomous-databases in compartment id
compartment-ocid
    where all {request.permission =  'AUTONOMOUS_DATABASE_CREATE',
target.autonomous-database.cloneType = /CLONE*/}
```

See Permissions for more information.

Policy Details for Autonomous Database

This topic covers details for writing policies to control access to Autonomous Database resources.

A policy defines what kind of access a group of users has to a specific resource in an individual compartment. For more information, see Getting Started with Policies.

Resource-Types

An aggregate resource-type covers the list of individual resource-types that directly follow. For example, writing one policy to allow a group to have access to the `autonomous-database-family` is equivalent to writing four separate policies for the group that would grant access to the `autonomous-databases`, `autonomous-backups` resource-types. For more information, see Resource-Types.

Resource-Types for Autonomous Database
Aggregate Resource-Type:

`autonomous-database-family`

Individual Resource-Types:

`autonomous-databases`

`autonomous-backups`

Supported Variables

General variables are supported. See General Variables for All Requests for more information.

Additionally, you can use the `target.workloadType` variable as shown in the following table:

target.workloadType Value	Description
`OLTP`	Online Transaction Processing, used for Autonomous Databases with Transaction Processing workload.
`DW`	Data Warehouse, used for Autonomous Databases with Data Warehouse workload.
`AJD`	Autonomous JSON Database used for Autonomous Databases with JSON workload.
`APEX`	APEX Service used for Autonomous Database APEX Service.

Example policy using the `target.workloadType` variable:

```
Allow group ADB-Admins to manage autonomous-databases in tenancy where
target.workloadType = 'AJD'
```

Details for Verb + Resource-Type Combinations

The level of access is cumulative as you go from `inspect > read > use > manage`. A plus sign (+) in a table cell indicates incremental access compared to the cell directly above it, whereas "no extra" indicates no incremental access.

For example, the `read` verb for the `autonomous-databases` resource-type covers the same permissions and API operations as the `inspect` verb, plus the `AUTONOMOUS_DATABASE_CONTENT_READ` permission. The `read` verb partially covers the `CreateAutonomousDatabaseBackup` operation, which also needs manage permissions for `autonomous-backups`.

The following tables show the Permissions and API operations covered by each verb. For information about permissions, see Permissions.

> **Note:**
>
> The resource family covered by **autonomous-database-family** can be used to grant access to database resources associated with all the Autonomous Database workload types.

autonomous-databases Resource Types

Verbs	Permissions	APIs Fully Covered	APIs Partially Covered
inspect	`AUTONOMOUS_DATABASE_INSPECT`	`GetAutonomousDatabase`, `ListAutonomousDatabases`	*none*
read	*`INSPECT`* + `AUTONOMOUS_DATABASE_CONTENT_READ`	*no extra*	`CreateAutonomousDatabaseBackup` (also needs `manage autonomous-backups`)

Verbs	Permissions	APIs Fully Covered	APIs Partially Covered
use	*READ* + AUTONOMOUS_DATABASE_CONTENT_WRITE AUTONOMOUS_DATABASE_UPDATE	UpdateAutonomousDatabase	RestoreAutonomousDatabase (also needs read autonomous-backups) ChangeAutonomousDatabaseCompartment (also needs read autonomous-backups)
manage	*USE* + AUTONOMOUS_DATABASE_CREATE AUTONOMOUS_DATABASE_DELETE	CreateAutonomousDatabase	*none*

autonomous-backups

Verbs	Permissions	APIs Fully Covered	APIs Partially Covered
inspect	AUTONOMOUS_DB_BACKUP_INSPECT	ListAutonomousDatabaseBackups, GetAutonomousDatabaseBackup	*none*
manage	*USE* + AUTONOMOUS_DB_BACKUP_CREATE AUTONOMOUS_DB_BACKUP_DELETE	DeleteAutonomousDatabaseBackup	CreateAutonomousDatabaseBackup (also needs read autonomous-databases)
read	*INSPECT* + AUTONOMOUS_DB_BACKUP_CONTENT_READ	*no extra*	RestoreAutonomousDatabase (also needs use autonomous-databases) ChangeAutonomousDatabaseCompartment (also needs use autonomous-databases)
use	*READ* + *no extra*	*no extra*	*none*

Policies to Manage Autonomous Databases

Provides a list of the IAM policies required for a cloud user to perform management operations on Autonomous Databases.

Operation	Required IAM Policies
Create a database	manage autonomous-databases read autonomous-databases
View a list of databases	inspect autonomous-databases
View details of a database	inspect autonomous-databases

Operation	Required IAM Policies
Set the password of a database's ADMIN user	`use autonomous-databases`
Scale the CPU core count or storage of a database	`use autonomous-databases`
Enable or disable auto scaling for a database	`use autonomous-databases`
Move a database to another compartment	`use autonomous-databases` in the database's current compartment and in the compartment you are moving it to `read autonomous-backups`
Stop or start a database	`use autonomous-databases`
Restart a database	`use autonomous-databases`
Back up a database manually	`read autonomous-databases` `manage autonomous-backups`
Restore a database	`use autonomous-databases` `read autonomous-backups`
Clone a database	`manage autonomous-databases` See IAM Permissions and API Operations for Autonomous Database for additional cloning permissions on Autonomous Database.
Terminate a database	`manage autonomous-databases`

Use Oracle Data Safe with Autonomous Database

Provides information on using Oracle Data Safe on Autonomous Database.

- About Oracle Data Safe with Autonomous Database
  Oracle Data Safe, which is included with Autonomous Database, provides a unified control center that helps you manage the day-to-day security and compliance requirements of Oracle Databases.
- Register Autonomous Database with Oracle Data Safe
  To use Oracle Data Safe you first need to register your database with Oracle Data Safe.
- Use Oracle Data Safe Features
  After you register Autonomous Database with Oracle Data Safe you can use the Data Safe features.

About Oracle Data Safe with Autonomous Database

Oracle Data Safe, which is included with Autonomous Database, provides a unified control center that helps you manage the day-to-day security and compliance requirements of Oracle Databases.

Data Safe helps you to evaluate security controls, assess user security, monitor user activity, mitigate risk from compromised accounts, and address data security compliance requirements for your database. Data Safe accomplishes this by evaluating the sensitivity of your data and assisting you when you need to mask sensitive data for non-production databases.

Oracle Data Safe provides features to assist you when:

- Your organization's policies require that you monitor your databases and retain audit records.

- You need to protect against common database attacks coming from risks such as compromised accounts.
- Your developers need to use copies of production data for work on a new application and you're wondering what kinds of sensitive information the production data contains.
- You need to make sure that staff changes haven't left dormant user accounts on your databases.

Oracle Data Safe provides the following:

- **Security Assessment**: Configuration errors and configuration drift are significant contributors to data breaches. Use security assessment to evaluate your database's configuration and compare it to Oracle and industry best practices. Security assessment provides reports on areas of risk and notifies you when configurations change.
- **User Assessment**: Many breaches start with a compromised user account. User Assessment helps you spot the riskiest database accounts, those accounts which if compromised could cause the most damage. User Assessment helps you take proactive steps to secure these accounts. User Assessment Baselines make it easy to know when new accounts are added, or when an account's privileges are modified. You can use Oracle Cloud Infrastructure Events to receive proactive notifications when a database deviates from its baseline.
- **Data Discovery**: Provides support to locate and to manage sensitive data in your applications. Data discovery scans your database for over 150 different types of sensitive data and helps you to understand what types and how much sensitive data you are storing. Use the data discovery reports to formulate audit policies, develop data masking templates, and create effective access control policies.
- **Data Masking** Minimize the amount of sensitive data your organization maintains to help you meet compliance requirements and satisfy data privacy regulations. Data masking helps you remove risk from your non-production databases by replacing sensitive information with masked data. With reusable masking templates, over 50 included masking formats, and the ability to easily create custom formats for your organization's unique requirements, data masking can streamline your application development and testing operations.
- **Activity Auditing** Activity auditing collects audit records from databases and helps you manage audit policies. Understanding and reporting on user activity, data access, and changes to database structures supports regulatory compliance requirements and can aid in post-incident investigations. Audit insights make it easy to identify inefficient audit policies, while alerts based on audit data proactively notify you of risky activity.

> **Note:**
>
> One (1) million audit records per database per month are included for your Autonomous Database if using the audit collection for Activity Auditing in Oracle Data Safe.

See Oracle Data Safe Overview for more information.

Register Autonomous Database with Oracle Data Safe

To use Oracle Data Safe you first need to register your database with Oracle Data Safe.

To get started, register your database:

1. Apply the necessary Identity and Access Management (IAM) permissions to register your target database.

   See Permissions to register an Autonomous Database for more information.

2. If you are registering an Autonomous Database that is configured to use a private IP address, then you need to create an Oracle Data Safe private endpoint either before or during registration.

   See Create an Oracle Data Safe Private Endpoint for more information.

3. Use the Oracle Data Safe Wizard to register your Autonomous Database instance.

   See Register an Autonomous Database for details on the registration steps.

Use Oracle Data Safe Features

After you register Autonomous Database with Oracle Data Safe you can use the Data Safe features.

Data Safe Feature	More Information
Security Assessment	Security Assessments are automatically scheduled once a week. Start by reviewing the security assessment report for your database: View the latest assessment for a target database. See Security Assessment Overview for more information.
User Assessment	User Assessments are automatically scheduled once a week. Start by reviewing the user assessment report for your database: View the latest user assessment for a target database See User Assessment Overview for more information.
Data Discovery	Start by discovering sensitive data in your database: Create Sensitive Data Models See Data Discovery Overview for more information.
Data Masking	To determine where sensitive data is stored in your database, run Data Discovery. After you know where sensitive data is stored in your database, you can create a masking policy: Create Masking Policies For example, after you create a masking policy you can make a copy of a production database and apply the masking policy to the non-production database: Mask Sensitive Data on a Target Database See Data Masking Overview for more information.
Activity Auditing	To use activity auditing, start the audit trail for your target database in Data Safe: Start an Audit Trail After the audit trail is started you can monitor and analyze your audit data with pre-defined audit reports: View a Predefined or Custom Audit Report See Activity Auditing Overview for more information.

Break Glass Access for SaaS on Autonomous Database

Autonomous Database supports break glass access for SaaS providers. Break glass access allows a SaaS operations team, when explicitly authorized by a SaaS customer, to access a customer's database to perform critical or emergency operations.

- About Break Glass Access on Autonomous Database
  Break glass access on Autonomous Database supports SaaS providers, where the SaaS organization defines procedures to permit a SaaS operations team member to access a customer's database when they are explicitly authorized by the customer.

- Enable Break Glass Access
  After authorization to access a database with `SAAS_ADMIN` is approved through procedures defined by your organization, use the Autonomous Database CLI or API to enable the `SAAS_ADMIN` user.
- Disable Break Glass Access
  Use the Autonomous Database CLI or API to disable `SAAS_ADMIN` user access.
- Notes for Break Glass Access
  Provides notes for break glass access.

About Break Glass Access on Autonomous Database

Break glass access on Autonomous Database supports SaaS providers, where the SaaS organization defines procedures to permit a SaaS operations team member to access a customer's database when they are explicitly authorized by the customer.

Break Glass Sample Use Case with Example.com

Consider a SaaS provider named `example.com` that is using Autonomous Database for their product. In usual operations the SaaS provider, `example.com`, creates an Autonomous Database instance for each SaaS customer. In this model a SaaS customer, for example a customer named Scott, is an end-user for the `example.com` product (and a SaaS customer whose data is stored in an Autonomous Database instance). The provider `example.com` creates and stores all of Scott's data in an Autonomous Database instance, and the customer, Scott, has no direct access to the database.

This SaaS model is summarized as follows:

- The Oracle customer creating Autonomous Database instances is the SaaS organization, (`example.com`).
- The SaaS provider is `example.com`.
- The SaaS customer is Scott.

If and when something goes wrong with regards to application performance, or there is some other critical issue that requires troubleshooting by the SaaS operations team, the customer Scott, can grant access so that the operations team can access Scott's database for troubleshooting. The SaaS operations team is only authorized to establish direct access to Scott's Autonomous Database instance through a SaaS defined approval process (in other words, after `example.com` receives permission from their customer, Scott).

Break Glass and the Autonomous Database SAAS_ADMIN User

When a SaaS invokes the break glass API on a customer's Autonomous Database instance, this enables the `SAAS_ADMIN` user. The SaaS operations team can then access the instance using the `SAAS_ADMIN` user with a specified set of roles, for a limited time.
By default the `SAAS_ADMIN` user is locked. Using an approval process defined by the SaaS organization, the `SAAS_ADMIN` user can be enabled to allow access to an Autonomous Database instance. The break glass name comes from manual fire alarms that require their users to break a small glass window pane before activating the alarm (the glass must be broken to prevent the alarm from being triggered by mistake). Similarly, the `SAAS_ADMIN` user doesn't normally access the database and access requires a predefined approval process.

Depending on the type of access granted, when enabled, the `SAAS_ADMIN` user can access the database to investigate issues or to make changes associated with an emergency or other unusual event. When break glass access expires or when access is explicitly disabled, the

SAAS_ADMIN account password/secrets are immediately rotated and SAAS_ADMIN user access is revoked. All the actions that the SAAS_ADMIN user performs are audited.

The SAAS_ADMIN user is enabled with one of three access types:

- read-only: provides read only access to the instance. This is the default access type and includes default roles: CREATE SESSION, SELECT ANY TABLE, SELECT ANY DICTIONARY, SELECT_CATALOG_ROLE.
- read/write: provides read/write access to the instance. The default roles for this type are: CREATE SESSION, SELECT ANY TABLE, SELECT ANY DICTIONARY, SELECT_CATALOG_ROLE, INSERT ANY TABLE, and UPDATE ANY TABLE.
- admin: provides admin access to the instance. The default roles for this type are: CREATE SESSION and PDB_DBA.

Break Glass API

The SAAS_ADMIN user is enabled and disabled only through the Command Line Interface (CLI) or using the Autonomous Database REST APIs.

For information about using the REST APIs and signing requests, see REST APIs and Security Credentials.

For information about SDKs, see Software Development Kits and Command Line Interface.

Use these APIs for Break Glass operations:

- To enable or disable SAAS_ADMIN, use configureSaasAdminUser.
- To check if the SAAS_ADMIN user is enabled use getSaasAdminUserStatus.

Enable Break Glass Access

After authorization to access a database with SAAS_ADMIN is approved through procedures defined by your organization, use the Autonomous Database CLI or API to enable the SAAS_ADMIN user.

Before you enable the SAAS_ADMIN user to access a database you must obtain values for the required parameters.

Parameter	Description
isEnabled	Specifies a Boolean value. Use TRUE to enable.
password	Specifies the password for the SAAS_ADMIN user. If you specify secretId you cannot specify password. The password you provide as a parameter must conform to the Autonomous Database password requirements. See About User Passwords on Autonomous Database for more information.
secretId	Specifies the value of a secret's Oracle Cloud Infrastructure Vault secret OCID. If you specify password you cannot specify secretId. See Overview of Vault for more information. The password you provide as a secret in the Oracle Cloud Infrastructure Vault must conform to the Autonomous Database password requirements. See About User Passwords on Autonomous Database for more information.
secretVersionNumber	Specifies the version number of the secret specified with the secretId. This parameter is optional. The default is to use the latest secret version. This parameter only applies when secretId is also specified.

Parameter	Description
`accessType`	One of: `read-only`, `read/write`, or `admin`. The default is `read-only`.
`duration`	Specifies the duration in hours, in the range of 1 hour to 24 hours. The default is 1 hour.

- Enable Break Glass Access with a Password
  Use the Autonomous Database CLI or API to enable `SAAS_ADMIN` with a password.
- Enable Break Glass Access with a Vault Secret
  Use the Autonomous Database CLI or API to enable `SAAS_ADMIN` with a `secretId`, when the secret is stored in Oracle Cloud Infrastructure Vault.

Enable Break Glass Access with a Password

Use the Autonomous Database CLI or API to enable `SAAS_ADMIN` with a password.

1. Use the API or the CLI and specify a value for the password to enable `SAAS_ADMIN` with a `password`.

   For example:

   ```
   POST https://dbaas-api.svc.ad3.us-phoenix-1/20160918/autonomousDatabases/
   ocid1.autonomousdatabase.oc1.phx.uniqueId/actions/configureSaasAdminUser

   {  "isEnabled": true,
      "password": password,
      "accessType": "READ_ONLY",
      "duration": 17
   }
   ```

   See configureSaasAdminUser for more information.

2. Verify that `SAAS_ADMIN` user is enabled.

   ```
   POST https://dbaas-api.svc.ad3.us-phoenix-1/20160918/autonomousDatabases/
   ocid1.autonomousdatabase.oc1.phx.uniqueId/actions/getSaasAdminUserStatus

   {    "isEnabled": true,
        "accessType": "READ_ONLY",
        "timeSaasAdminUserEnabled": "2023-11-23T01:59:07.032Z"
   }
   ```

   This response shows that the `SAAS_ADMIN` user is enabled and that access type is `READ_ONLY`. The timestamp shows the time when `SAAS_ADMIN` was enabled (time is in UTC).

   See getSaasAdminUserStatus for more information.

Enable Break Glass Access with a Vault Secret

Use the Autonomous Database CLI or API to enable `SAAS_ADMIN` with a `secretId`, when the secret is stored in Oracle Cloud Infrastructure Vault.

When you specify a `secretId`, in order for Autonomous Database to reach the secret in the Oracle Cloud Infrastructure Vault, the following conditions must apply:

- The secret must be in `current` or `previous` state.
- You must have the proper user group policy that allows `READ` access to the specific secret in a given compartment. For example:

```
Allow userGroup1 to read secret-bundles in compartment training
```

To enable `SAAS_ADMIN` with a `secretId` with the secret stored in Oracle Cloud Infrastructure Vault:

1. Use the API or the CLI and specify an OCID value for the `secretId`.

   For example:

```
POST https://dbaas-api.svc.ad3.us-phoenix-1/20160918/autonomousDatabases/
ocid1.autonomousdatabase.oc1.phx.uniqueId/actions/configureSaasAdminUser

{  "isEnabled": true,
   "secretId": "ocid1.key.co1.ap-
mumbai-1.example..aaaaaaaauq5ok5nq3bf2vwetkpqsoa",
   "accessType": "READ_ONLY",
   "duration": 20
}
```

   Specifying a secret version is optional. If you specify a secret version in the API call with `secretVersionNumber`, the specified secret version is used. If you do not specify a secret version the call uses the latest secret version.

   See configureSaasAdminUser for more information.

2. Verify that `SAAS_ADMIN` user is enabled.

   For example:

```
POST https://dbaas-api.svc.ad3.us-phoenix-1/20160918/autonomousDatabases/
ocid1.autonomousdatabase.oc1.phx.uniqueId/actions/getSaasAdminUserStatus

{    "isEnabled": true,
     "accessType": "READ_ONLY",
     "timeSaasAdminUserEnabled": "2023-11-23T01:59:07.032Z"
}
```

   This response shows that the `SAAS_ADMIN` user is enabled and the access type is `READ_ONLY`. The timestamp shows the time when the user was enabled (time is in UTC).

   See getSaasAdminUserStatus for more information.

Disable Break Glass Access

Use the Autonomous Database CLI or API to disable `SAAS_ADMIN` user access.

By default access expires after one hour if the `duration` parameter is not set when `SAAS_ADMIN` is enabled. If the `duration` parameter is set when `SAAS_ADMIN` is enabled, then access expires after the specified `duration` number of hours. As an alternative to letting the access expire based on the either the default expiration time or the duration you specify, you can use configureSaasAdminUser to explicitly disable `SAAS_ADMIN` user access.

1. Disable `SAAS_ADMIN` user access.

   For example:

   ```
   POST https://dbaas-api.svc.ad3.us-phoenix-1/20160918/autonomousDatabases/
   ocid1.autonomousdatabase.oc1.phx.uniqueId/actions/configureSaasAdminUser
   {
     "isEnabled": false
   }
   ```

   See configureSaasAdminUser API for more information.

2. Verify that `SAAS_ADMIN` user is disabled.

   For example:

   ```
   POST https://dbaas-api.svc.ad3.us-phoenix-1/20160918/autonomousDatabases/
   ocid1.autonomousdatabase.oc1.phx.uniqueId/actions/getSaasAdminUserStatus

   {    "isEnabled": false
   }
   ```

   This response indicates that the `SAAS_ADMIN` user is disabled.

   See getSaasAdminUserStatus for more information.

When you disable the `SAAS_ADMIN` user, access to the database is revoked and Autonomous Database rotates the password or the secret that was used to access the database.

Notes for Break Glass Access

Provides notes for break glass access.

Notes for break glass access:

- The `duration` you specify when you enable `SAAS_ADMIN` applies either until the specified time expires or until you explicitly disable the `SAAS_ADMIN` user. You cannot change this value after you enable the `SAAS_ADMIN` user.
- Always Free Autonomous Database does not support access with the `SAAS_ADMIN` user.
- The `DBA_CLOUD_CONFIG` view provides information on the `SAAS_ADMIN` user.

  For example, the use the following query to obtain information on the status for the `SAAS_ADMIN` user:

  ```
  SELECT PARAM_VALUE FROM DBA_CLOUD_CONFIG WHERE
              param_name ='saas_admin_access' order by 1;
  ```

  The presence of a value for `auth_revoker` means the access has been revoked and shows the user who revoked access.

  The `auth_end` shows a `planned` time. After authorization is revoked, if the authorization expired at the time end of the `duration` specified when `SAAS_ADMIN` user was enabled, the `planned` time will be the same as the `actual` time. If the `planned` and the `actual` time are different, this means that `SAAS_ADMIN` authorization was revoked before the `duration` expired.

For example, if `SAAS_ADMIN` is enabled with a duration of 2 hours, and after 1 hour access is disabled by calling the API configureSaasAdminUser to disable the `SAAS_ADMIN` user, the `auth_end planned` and `actual` times will be different.

If this query shows no rows, then the `SAAS_ADMIN` user does not exist. It may have been created and dropped or it was never created.

Encrypt Data While Exporting or Decrypt Data While Importing

For greater security you can encrypt data that you export to Object Storage. When data on Object Storage is encrypted you can decrypt the data you import or when you use the data in an external table.

- Encrypt Data While Exporting to Object Storage
  You can encrypt table data while exporting to Object Storage.
- Decrypt Data While Importing from Object Storage
  You can decrypt and load data from encrypted files stored in Object Storage. You can also decrypt encrypted data on Object Storage that you use in an external table.

Encrypt Data While Exporting to Object Storage

You can encrypt table data while exporting to Object Storage.

Use the `format` parameter and the `encryption` option with `DBMS_CLOUD.EXPORT_DATA` to encrypt data when you export from Autonomous Database to Object Storage.

Note the following when you export encrypted data to Object Storage:

- The `encryption` option is only supported when exporting data from Autonomous Database to Object Storage as CSV, JSON, or XML.
- When the export includes both encryption and compression, the order of operations is: first the data is compressed, next the data is encrypted, and then it is uploaded to Object Storage.
- There are two supported encryption methods:
  - Using a user-defined function.
  - Using a `DBMS_CRYPTO` specified encryption algorithm.

    See DBMS_CRYPTO for information on the cryptographic functions and procedures for encryption and decryption.

Topics

- Encrypt Data Using DBMS_CRYPTO Encryption Algorithms
  Shows the steps to encrypt data using `DBMS_CRYPTO` encryption algorithms while exporting to Cloud Object Storage.
- Encrypt Data with a User Defined Encryption Function
  Shows the steps to encrypt data using a user-defined encryption function while exporting to Cloud Object Storage.

Encrypt Data Using DBMS_CRYPTO Encryption Algorithms

Shows the steps to encrypt data using `DBMS_CRYPTO` encryption algorithms while exporting to Cloud Object Storage.

Perform the following steps to encrypt data while exporting to Cloud Object Storage (this example exports table data to a CSV file):

1. Connect to your Autonomous Database instance.

   See Connect to Autonomous Database for more information.

2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'OBJ_STORE_CRED',
       username        => 'user1@example.com',
       password        => 'password'
     );
   END;
   /
   ```

   The values you provide for `username` and `password` depend on the Cloud Object Storage service you are using.

   See CREATE_CREDENTIAL Procedure for more information.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Create a credential to store the encryption key (the encryption key to be used for encrypting data).

   When you encrypt data using `DBMS_CRYPTO` encryption algorithms you store the encryption key in a credential. The key is specified in the `password` field in a credential you create with `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL (
       credential_name => 'ENC_CRED_NAME',
       username        => 'Any_username',
       password        => 'password'
     );
   END;
   /
   ```

   As an alternative you can create a credential to store the key in a vault. For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL (
       credential_name  => 'ENC_CRED_NAME',
       params           => JSON_OBJECT ('username' value 'Any_username',
   ```

```
                                        'region'    value 'Region',
                                        'secret_id' value 'Secret_id_value'));
END;
/
```

> **Note:**
>
> The `username` parameter you specify in the credential that stores the key can be any string.

This creates the `ENC_CRED_NAME` credential which is a vault secret credential, where the secret (decryption/encryption key) is stored as a secret in Oracle Cloud Infrastructure Vault.

See CREATE_CREDENTIAL Procedure for more information.

4. Run `DBMS_CLOUD.EXPORT_DATA`.

   Use the `format` parameter with the `encryption` option. The `encryption` type specifies the `DBMS_CRYPTO` encryption algorithm to use to encrypt the table data and the `credential_name` value is credential that specifies the secret (encryption key).

   For example:

```
BEGIN
    DBMS_CLOUD.EXPORT_DATA (
        credential_name => 'OBJ_STORE_CRED',
        file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namepace-string/b/bucketname/o/encrypted.csv',
        query           => 'SELECT * FROM ADMIN.employees',
        format          => json_object(
               'type' value 'csv',
               'encryption' value  json_object(
                      'type' value DBMS_CRYPTO.ENCRYPT_AES256 +
DBMS_CRYPTO.CHAIN_CBC + DBMS_CRYPTO.PAD_PKCS5,
                      'credential_name' value 'ENC_CRED_NAME'))
      );
END;
/
```

   This encrypts and exports the data from the `EMPLOYEES` table into a CSV file.

   See DBMS_CRYPTO Algorithms for more information on encryption algorithms.

   In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

   See EXPORT_DATA Procedure and DBMS_CLOUD Package Format Options for EXPORT_DATA for more information.

After you encrypt files with `DBMS_CLOUD.EXPORT_DATA`, when you use `DBMS_CRYPTO` encryption algorithms to encrypt the files, you have these options for using or importing the files you exported:

- You can use `DBMS_CLOUD.COPY_DATA` or `DBMS_CLOUD.COPY_COLLECTION` with the same encryption algorithm options and the key to decrypt the files.

See Decrypt and Load Data Using DBMS_CRYPTO Algorithms for more information.

- You can query the data in an external table by supplying the same encryption algorithm options and the key to decrypt the files, with any of the following procedures:
  - `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`
  - `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`
  - `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE`

    For `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` this option is only applicable to the Object Storage files.

  See Decrypt and Load Data Using DBMS_CRYPTO Algorithms for more information.
- On a system that is not an Autonomous Database you can use the `DBMS_CRYPTO` package with the same algorithm options and the key to decrypt the files.

  Note that the key is stored as a `VARCHAR2` in the credential in Autonomous Database but `DBMS_CRYPTO` uses `RAW` type for the key parameter.

  See DBMS_CRYPTO Algorithms for more information on encryption algorithms.

Encrypt Data with a User Defined Encryption Function

Shows the steps to encrypt data using a user-defined encryption function while exporting to Cloud Object Storage.

Perform the following steps to encrypt data while exporting to Cloud Object Storage (this example exports table data to a CSV file):

1. Connect to your Autonomous Database instance.

   See Connect to Autonomous Database for more information.
2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'OBJ_STORE_CRED',
    username        => 'user1@example.com',
    password        => 'password'
  );
END;
/
```

   The values you provide for `username` and `password` depend on the Cloud Object Storage service you are using.

   See CREATE_CREDENTIAL Procedure for more information.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.
3. Create a user-defined callback function to encrypt data.

For example:

```
CREATE OR REPLACE FUNCTION encryption_func (data IN BLOB)
  RETURN BLOB
  IS
      l_encrypted_data BLOB;
       BEGIN
     DBMS_LOB.CREATETEMPORARY (l_encrypted_data, TRUE, DBMS_LOB.CALL);
     DBMS_CRYPTO.ENCRYPT (
         dst => l_encrypted_data,
         src => data,
         typ => DBMS_CRYPTO.ENCRYPT_AES256 + DBMS_CRYPTO.CHAIN_CBC +
DBMS_CRYPTO.PAD_PKCS5,
         key => 'encryption key'
      );
     RETURN l_encrypted_data;
END encryption_func;
/
```

This creates the `ENCRYPTION_FUNC` encryption function. This function encrypts data using a stream or block cipher with a user supplied key.

> **Note:**
>
> You must create an encryption key to be used as a value in the `KEY` parameter. See DBMS_CRYPTO Operational Notes for more information on generating the encryption key.

4. Run `DBMS_CLOUD.EXPORT_DATA` with the `format` parameter, include the `encryption` option and specify a `user_defined_function`.

   For example:

```
BEGIN
      DBMS_CLOUD.EXPORT_DATA (
        credential_name => 'OBJ_STORE_CRED',
        file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namepace-string/b/bucketname/o/encrypted.csv',
        query           => 'SELECT * FROM ADMIN.emp',
        format          => json_object(
                               'type' value 'csv',
                               'encryption' value
json_object('user_defined_function' value 'admin.encryption_func'))
      );
END;
/
```

This encrypts the data from the specified query the on `EMP` table and exports the data as a CSV file on Cloud Object Storage. The `format` parameter with the `encryption` value specifies the user-defined encryption function to use to encrypt the data.

> **Note:**
>
> You must have `EXECUTE` privilege on the encryption function.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

See EXPORT_DATA Procedure and DBMS_CLOUD Package Format Options for EXPORT_DATA for more information.

Decrypt Data While Importing from Object Storage

You can decrypt and load data from encrypted files stored in Object Storage. You can also decrypt encrypted data on Object Storage that you use in an external table.

This option is useful when migrating from an on-premises database to an Autonomous Database if the data in your source files is encrypted.

> **Note:**
>
> This option is only supported for Object Storage files less than 4 GB.

This option is applicable for the following procedures:

- `DBMS_CLOUD.COPY_DATA`
- `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`
- `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`
- `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE`

  For `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` this option is only applicable to the Object Storage files.
- `DBMS_CLOUD.COPY_COLLECTION`

Topics

- Decrypt and Load Data Using DBMS_CRYPTO Algorithms
  Shows the steps to decrypt encrypted files from Object Storage and load the data into a table on Autonomous Database (the decrypt step uses `DBMS_CRYPTO` algorithms).
- Decrypt and Load Data with a User Defined Function
  Shows the steps to decrypt files in Object Storage and load the data into tables using a user-defined decryption function.

Decrypt and Load Data Using DBMS_CRYPTO Algorithms

Shows the steps to decrypt encrypted files from Object Storage and load the data into a table on Autonomous Database (the decrypt step uses `DBMS_CRYPTO` algorithms).

As a prerequisite you must have encrypted files and uploaded the files into Object Storage. This example uses a CSV file and it is assumed that the file is encrypted using

`DBMS_CRYPTO.ENCRYPT_AES256 + DBMS_CRYPTO.CHAIN_CBC + DBMS_CRYPTO.PAD_PKCS5` algorithm and uploaded to your Cloud Object Storage.

See ENCRYPT Function for more information on the `ENCRYPT` function.

See DBMS_CRYPTO Operational Notes for more information on generating an encryption key.

To decrypt and load data into an existing table on Autonomous Database from Object Storage:

1. Connect to your Autonomous Database instance.

   See Connect to Autonomous Database for more information.

2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'OBJ_STORE_CRED',
       username        => 'user1@example.com',
       password        => 'password'
     );
   END;
   /
   ```

   The values you provide for `username` and `password` depend on the Cloud Object Storage service you are using.

   See CREATE_CREDENTIAL Procedure for more information.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Create a credential to store the key using `DBMS_CLOUD.CREATE_CREDENTIAL`. For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL (
       credential_name => 'ENC_CRED_NAME',
       username        => 'Any_username',
       password        => 'password'
     );
   END;
   /
   ```

   As an alternative you can create credentials to store the key in a vault. For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL (
       credential_name  => 'ENC_CRED_NAME',
       params           => JSON_OBJECT ('username' value 'Any_username',
                                        'region'    value 'Region',
                                        'secret_id' value 'Secret_id_value'));
   END;
   /
   ```

> **Note:**
>
> The `username` parameter you specify in the credential that stores the key can be any string.

This creates the `ENC_CRED_NAME` credential which is a vault secret credential, where the secret (decryption/encryption key) is stored as a secret in the Oracle Cloud Infrastructure Vault.

See CREATE_CREDENTIAL Procedure for more information.

4. Run `DBMS_CLOUD.COPY_DATA` and specify `DBMS_CRYPTO` encryption algorithm as decryption method.

```
BEGIN
 DBMS_CLOUD.COPY_DATA (
   table_name       => 'CSV_COPY_DATA',
   credential_name  => 'OCI$RESOURCE_PRINCIPAL',
   file_uri_list    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namepace-string/b/bucketname/o/encrypted.csv',
   format           => json_object( 'type' value 'csv', 'encryption' value
json_object('type' value DBMS_CRYPTO.ENCRYPT_AES256 +
DBMS_CRYPTO.CHAIN_CBC + DBMS_CRYPTO.PAD_PKCS5,'credential_name' value
'ENC_CRED_NAME'))
  );
END;
/
```

This decrypts the `ENCRYPTED.CSV` file in the Object Storage. The data is then loaded into the `CSV_COPY_DATA` table. The `format` parameter `encryption` option value specifies a `DBMS_CRYPTO` encryption algorithm to be used to decrypt data.

See DBMS_CRYPTO Algorithms for more information on encryption algorithms.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

For detailed information about the parameters, see COPY_DATA Procedure.

For detailed information about the available `format` parameters, you can use with `DBMS_CLOUD.COPY_DATA`, see DBMS_CLOUD Package Format Options.

Decrypt and Load Data with a User Defined Function

Shows the steps to decrypt files in Object Storage and load the data into tables using a user-defined decryption function.

As a prerequisite for these steps you must have encrypted files and uploaded the files into Object Storage. This example uses a CSV file and it is assumed that the file is encrypted using `DBMS_CRYPTO.ENCRYPT_AES256 + DBMS_CRYPTO.CHAIN_CBC + DBMS_CRYPTO.PAD_PKCS5` algorithm and uploaded to your Cloud Object Storage.

See ENCRYPT Function for more information on the `ENCRYPT` function.

See DBMS_CRYPTO Operational Notes for more information on generating an encryption key.

To decrypt and load data into an existing table on Autonomous Database from Object Storage:

1. Connect to your Autonomous Database instance.

   See Connect to Autonomous Database for more information.

2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'OBJ_STORE_CRED',
    username        => 'user1@example.com',
    password        => 'password'
  );
END;
/
```

   The values you provide for `username` and `password` depend on the Cloud Object Storage service you are using.

   See CREATE_CREDENTIAL Procedure for more information.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Create a user-defined function decryption callback function.

   For example:

```
CREATE OR REPLACE FUNCTION decryption_func (data IN BLOB)
RETURN BLOB
IS
  l_decrypted_data     BLOB;
  response             DBMS_CLOUD_TYPES.resp;
  l_response_json      JSON_ARRAY_T;
  l_response_json_obj  JSON_OBJECT_T;
  l_secret_id          VARCHAR2(4000) := '<secret_ocid>';
  l_key                VARCHAR2(4000);
BEGIN
   response := DBMS_CLOUD.send_request (
               credential_name => 'OCI$RESOURCE_PRINCIPAL',
               uri             => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namepace-string/b/bucketname/o/' ||
l_secret_id,
               method          => 'GET');
  l_response_json_obj := JSON_OBJECT_T.parse
(DBMS_CLOUD.get_response_text(response));
  l_key := UTL_RAW.cast_to_varchar2
(UTL_ENCODE.base64_decode(UTL_RAW.cast_to_raw
(l_response_json_obj.get_object('secretBundleContent').get_string('content'
))));

  DBMS_LOB.createtemporary (l_decrypted_data, TRUE, DBMS_LOB.CALL);
  DBMS_CRYPTO.decrypt (
      dst => l_decrypted_data,
      src => data,
```

```
      typ => DBMS_CRYPTO.ENCRYPT_AES256 + DBMS_CRYPTO.CHAIN_CBC +
DBMS_CRYPTO.PAD_PKCS5,
      key => l_key
    );
  RETURN l_decrypted_data;
END decryption_func;
/
```

This creates the `DECRYPTION_FUNC` decryption function. This function decrypts data using a stream or block cipher with a user supplied key. The user supplied key in the example is stored in Oracle Cloud Infrastructure Vault and is retrieved dynamically by making a REST call to Oracle Cloud Infrastructure Vault service.

4. Run `DBMS_CLOUD.COPY_DATA` and specify the `format` option `encryption` and specify the user-defined function you created to decrypt the data.

```
BEGIN
 DBMS_CLOUD.COPY_DATA (
    table_name      => 'CSV_COPY_DATA',
    credential_name => 'OBJ_STORE_CRED',
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namepace-string/b/bucketname/o/encrypted.csv',
    format          => json_object(
                         'type' value 'csv',
                         'encryption' value
json_object('user_defined_function' value 'admin.decryption_func'))
  );
end;
/
```

This decrypts the `ENCRYPTED.CSV` file in the Object Storage. The data is then loaded into the `CSV_COPY_DATA` table. The `format` parameter `encryption` option value specifies a user-defined function name to use to decrypt data.

> **Note:**
>
> You must have the `EXECUTE` privilege on the user-defined function.

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

For detailed information about the parameters, see COPY_DATA Procedure.

For detailed information about the available `format` parameters, you can use with `DBMS_CLOUD.COPY_DATA`, see DBMS_CLOUD Package Format Options.

Manage Oracle Cloud Operator Access

Describes how to grant temporary access to your database for an Oracle Cloud Operator.

Oracle Cloud Operators do not have authorization to access your data or any other information in your database schemas. When access to your database schemas is required to

troubleshoot or mitigate an issue, you can allow a cloud operator to access your Autonomous Database schemas for a limited time.

You allow a cloud operator to access the database schemas by running the procedure `DBMS_CLOUD_ADMIN.ENABLE_OPERATOR_ACCESS`. This means if you file a service request with Oracle Cloud Support and Oracle Cloud Operators need to access your database schemas, you must also enable operator access by running `DBMS_CLOUD_ADMIN.ENABLE_OPERATOR_ACCESS`.

Topics

- Enable Cloud Operator Access
  Describes the steps to enable Cloud Operator access on an Autonomous Database instance.
- Disable Cloud Operator Access
  Describes the steps to disable Cloud Operator access on an Autonomous Database instance.

Enable Cloud Operator Access

Describes the steps to enable Cloud Operator access on an Autonomous Database instance.

1. Enable operator access.

   For example:

   ```
   BEGIN
      DBMS_CLOUD_ADMIN.ENABLE_OPERATOR_ACCESS(
          auth_duration => 12 );
   END;
   /
   ```

   The `auth_duration` parameter specifies the number of Hours for which the Cloud Operator is granted access. The value must be a whole number in the range of 1 to 24 and the default is 1 (Hour).

   See ENABLE_OPERATOR_ACCESS Procedure for more information.

2. Verify that operator access is enabled.

   For example:

   ```
   SELECT param_name, param_value FROM DBA_CLOUD_CONFIG
         WHERE LOWER(param_name) = 'operator_access';


   PARAM_NAME PARAM_VALUE
   ---------------
   ------------------------------------------------------------------------
   -
   operator_access {"auth_grantor":"\"ADMIN\"","auth_begin":"26-FEB-24
   07.34.37.144846 PM UTC",
   "auth_duration":"12 hour", "planned_auth_end":"27-FEB-24 07.34.37.146297
   AM UTC"}
   ```

`DBMS_CLOUD_ADMIN.ENABLE_OPERATOR_ACCESS` allows access only to the Cloud Operator and does not enable access for any other user.

All operations performed by the Cloud Operator are stored in the view `DBA_OPERATOR_ACCESS`. See View Oracle Cloud Infrastructure Operations Actions for more information.

Disable Cloud Operator Access

Describes the steps to disable Cloud Operator access on an Autonomous Database instance.

1. Disable operator access.

   For example

   ```
   BEGIN
       DBMS_CLOUD_ADMIN.DISABLE_OPERATOR_ACCESS;
   END;
   /
   ```

   See DISABLE_OPERATOR_ACCESS Procedure for more information.

2. Verify that operator access is disabled.

   For example:

   ```
   SELECT param_name, param_value FROM DBA_CLOUD_CONFIG
         WHERE LOWER(param_name) = 'operator_access';


   No data found
   ```

Performance Monitor and Management

Oracle Autonomous Database Serverless includes several features that automatically monitor, analyze and optimize the performance of your Autonomous Database.

- Use Operations Insights on Autonomous Database
  Operations Insights is a cloud-native service that enables users to make informed, data-driven, Autonomous Database resource and performance management decisions.
- Manage Optimizer Statistics on Autonomous Database
  Describes Autonomous Database commands to run when you need to gather optimizer statistics or enable optimizer hints.
- Manage Automatic Indexing on Autonomous Database
  Automatic indexing automates the index management tasks in Autonomous Database. Auto Indexing is disabled by default in Autonomous Database.
- Manage Automatic Partitioning on Autonomous Database

Use Operations Insights on Autonomous Database

Operations Insights is a cloud-native service that enables users to make informed, data-driven, Autonomous Database resource and performance management decisions.

See About Oracle Cloud Infrastructure Operations Insights for details on Operations Insights.

See To enable or disable Operations Insights on an Autonomous Database for information about enabling and disabling Operations Insights.

Manage Optimizer Statistics on Autonomous Database

Describes Autonomous Database commands to run when you need to gather optimizer statistics or enable optimizer hints.

There are differences in the commands to run to gather optimizer statistics or enable optimizer hints, depending on your workload: Data Warehouse, Transaction Processing, or JSON Database.

- Manage Optimizer Statistics and Hints with Data Warehouse Workloads
  Describes Autonomous Database commands to run when you need to gather optimizer statistics or enable optimizer hints with Data Warehouse workloads.
- Manage Optimizer Statistics and Hints with Transaction Processing and JSON Database Workloads
  Describes Autonomous Database commands to run when you need to gather optimizer statistics or enable optimizer hints.

Manage Optimizer Statistics and Hints with Data Warehouse Workloads

Describes Autonomous Database commands to run when you need to gather optimizer statistics or enable optimizer hints with Data Warehouse workloads.

Manage Optimizer Statistics with Data Warehouse Workloads

Autonomous Database with Data Warehouse workloads gathers optimizer statistics automatically for tables loaded with direct path operations issued in SQL (direct path load operations that bypass the SQL data processing, such as SQL*Loader direct path, do not collect statistics). For example, for loads using the `DBMS_CLOUD` package the database gathers optimizer statistics automatically.

If you have tables modified using conventional DML operations you can run commands to gather optimizer statistics for those tables. For example, for the `SH` schema you can gather statistics for all tables in the schema using the following command:

```
BEGIN
  DBMS_STATS.GATHER_SCHEMA_STATS('SH', options=>'GATHER AUTO');
END;
/
```

This example gathers statistics for all tables that have stale statistics in the `SH` schema.

For more information about direct-path loads see *Loading Tables*.

For more information on optimizer statistics see *Database Concepts*.

Manage Optimizer Hints with Data Warehouse Workloads

Autonomous Database with Data Warehouse ignores optimizer hints and `PARALLEL` hints in SQL statements by default. If your application relies on hints you can enable optimizer hints by setting the parameter `OPTIMIZER_IGNORE_HINTS` to `FALSE` at the session or system level using `ALTER SESSION` or `ALTER SYSTEM`. For example, the following command enables hints in your session:

```
ALTER SESSION
   SET OPTIMIZER_IGNORE_HINTS=FALSE;
```

You can also enable `PARALLEL` hints in your SQL statements by setting `OPTIMIZER_IGNORE_PARALLEL_HINTS` to `FALSE` at the session or system level using `ALTER SESSION` or `ALTER SYSTEM`. For example, the following command enables `PARALLEL` hints in your session:

```
ALTER SESSION
   SET OPTIMIZER_IGNORE_PARALLEL_HINTS=FALSE;
```

Manage Optimizer Statistics and Hints with Transaction Processing and JSON Database Workloads

Describes Autonomous Database commands to run when you need to gather optimizer statistics or enable optimizer hints.

Manage Optimizer Statistics with Transaction Processing and JSON Database Workloads

Autonomous Database gathers optimizer statistics automatically so that you do not need to perform this task manually and this helps to ensure your statistics are current. Automatic statistics gathering is enabled in Autonomous Database and runs in a standard maintenance window.

> **Note:**
>
> The automatic statistics gathering maintenance window is different than the maintenance window on the Oracle Cloud Infrastructure console. The Oracle Cloud Infrastructure maintenance window shows system patching information.

For more information on automatic statistics gathering maintenance window times and automatic optimizer statistics collection, see *Database Administrator's Guide*.

For more information on optimizer statistics see *SQL Tuning Guide*.

Manage Optimizer Hints with Transaction Processing and JSON Database Workloads

Autonomous Database with Transaction Processing and JSON Database workloads honors optimizer hints and `PARALLEL` hints in SQL statements by default. You can disable optimizer hints by setting the parameter `OPTIMIZER_IGNORE_HINTS` to `TRUE` at the session or system level using `ALTER SESSION` or `ALTER SYSTEM`. For example, the following command disables hints in your session:

```
ALTER SESSION
   SET OPTIMIZER_IGNORE_HINTS=TRUE;
```

You can also disable `PARALLEL` hints in your SQL statements by setting `OPTIMIZER_IGNORE_PARALLEL_HINTS` to `TRUE` at the session or system level using `ALTER SESSION` or `ALTER SYSTEM`.

```
ALTER SESSION
   SET OPTIMIZER_IGNORE_PARALLEL_HINTS=TRUE;
```

Manage Automatic Indexing on Autonomous Database

Automatic indexing automates the index management tasks in Autonomous Database. Auto Indexing is disabled by default in Autonomous Database.

Creating indexes manually requires deep knowledge of the data model, application, and data distribution. In the past, DBAs were responsible for making choices about which indexes to create, and then sometimes the DBAs did not revise their choices or maintain indexes as the conditions changed. As a result, opportunities for improvement were lost, and use of unnecessary indexes could be a performance liability. The automatic indexing feature in Autonomous Database monitors the application workload and creates and maintains indexes automatically.

To enable automatic indexing:

1. Use the `DBMS_AUTO_INDEX.CONFIGURE` procedure to enable automatic indexing:

   ```
   EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE','IMPLEMENT');
   ```

   This enables automatic indexing in a database and creates any new auto indexes as visible indexes, so that they can be used in SQL statements.
2. Use the `DBMS_AUTO_INDEX` package to report on the automatic task and to set automatic indexing preferences.

> **Note:**
>
> When automatic indexing is enabled, index compression for auto indexes is enabled by default.

To disable automatic indexing:

1. The following statement disables automatic indexing in a database so that no new auto indexes are created (existing auto indexes remain enabled):

   ```
   EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE','OFF');
   ```

When you use SODA with Autonomous Database the following restrictions apply:

- Automatic indexing is not supported for SQL and PL/SQL code that uses the SQL/JSON function `json_exists`. See SQL/JSON Condition JSON_EXISTS for more information.
- Automatic indexing is not supported for SODA query-by-example (QBE).

See Managing Auto Indexes for more information.

Manage Automatic Partitioning on Autonomous Database

Automatic partitioning analyzes and automates partition creation for tables and indexes of a specified schema to improve performance and manageability in Autonomous Database.

Automatic partitioning, when applied, is transparent and does not require any user interaction or maintenance.

> **Note:**
>
> Automatic partitioning does not interfere with existing partitioning strategies and is complementary to manual partitioning in Autonomous Database. Manually partitioned tables are excluded as candidates for automatic partitioning.

- About Automatic Partitioning
  Automatic partitioning in Autonomous Database analyzes the application workload and automatically applies partitioning to tables and their indexes to improve performance or to allow better management of large tables.
- How Automatic Partitioning Works
- Configure Automatic Partitioning
  Use the `DBMS_AUTO_PARTITION.CONFIGURE` procedure to configure automatic partitioning options.
- Use Automatic Partitioning
  Describes the flow and general processes for using and managing automatic partitioning in Autonomous Database.
- Generate Automatic Partitioning Reports
  Generate automatic partitioning reports using the `REPORT_ACTIVITY` and `REPORT_LAST_ACTIVITY` functions of the `DBMS_AUTO_PARTITION` package.
- Example Automatic Partitioning Scenarios
  Example scenarios for automatic partitioning using the `DBMS_AUTO_PARTITION` API procedures and functions.
- Data Dictionary Views for Automatic Partitioning
  There are two new views and one updated view in the data dictionary for information about the automatic partitioning configuration and recommendations in your database.

About Automatic Partitioning

Automatic partitioning in Autonomous Database analyzes the application workload and automatically applies partitioning to tables and their indexes to improve performance or to allow better management of large tables.

Finding appropriate partitioning strategies requires deep knowledge of the application workload and the data distribution. When you perform manual partitioning, you must analyze your workload and make choices about how to apply partitioning to tables and indexes to improve the performance of applications. Automatic partitioning enables Autonomous Database users to benefit from partitioning without performing manual schema and workload analysis.

Automatic partitioning uses a single-column partition key combined with single-level partitioning. Automatic partitioning does not support more complex partitioning strategies such as multi-column partitioned tables or composite partitioning.

Automatic partitioning chooses from the following partition methods:

- AUTOMATIC INTERVAL: This choice is best suited for ranges of partition key values.
- LIST AUTOMATIC: This partitioning method applies to distinct partition key values.
- HASH: Applies partitioning on the partition key's hash values.

See Partitioning Concepts for more information.

Automatic partitioning provides the following functionality:

- Analyzes application workload and finds the optimal partitioning strategy to improve query performance for tables eligible for automatic partitioning.
- Provides PL/SQL APIs for configuring automatic partitioning in a database, generating reports about partitioning findings, and generating and applying an identified partitioning strategy for a given workload.

> **Note:**
>
> Automatic partitioning requires explicit calls to the `DBMS_AUTO_PARTITION` PL/SQL APIs to recommend and apply automatic partitioning to an Autonomous Database.

How Automatic Partitioning Works

Unlike automatic indexing, automatic partitioning does not run periodically as a background task. Automatic partitioning only runs when you invoke it using the `DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD` function.

When invoked, automatic partitioning identifies candidate tables for automatic partitioning, evaluates partition schemes, and implements a partitioning strategy.

When you invoke automatic partitioning it performs the following tasks:

1. Identifies candidate tables for automatic partitioning by analyzing the workload for selected candidate tables.

   By default, automatic partitioning uses the workload information collected in an Autonomous Database for analysis. Depending on the size of the workload, a sample of queries might be considered.
2. Evaluates partition schemes based on workload analysis, quantification, and verification of the performance benefits:
   a. Candidate empty partition schemes with synthesized statistics are created internally and analyzed for performance.
   b. The candidate scheme with the highest estimated IO reduction is chosen as the optimal partitioning strategy and is internally implemented to test and verify performance.
   c. If a candidate partition scheme does not improve performance beyond specified performance and regression criteria, automatic partitioning is not recommended.
3. Implements the optimal partitioning strategy, if configured to do so, for the tables analyzed by the automatic partitioning procedures.

Configure Automatic Partitioning

Use the `DBMS_AUTO_PARTITION.CONFIGURE` procedure to configure automatic partitioning options.

Enable and implement recommendations

```
EXEC DBMS_AUTO_PARTITION.CONFIGURE('AUTO_PARTITION_MODE','IMPLEMENT');
```

Enable recommendations, but do not implement those recommendations

```
EXEC DBMS_AUTO_PARTITION.CONFIGURE('AUTO_PARTITION_MODE','REPORT ONLY');
```

Disable new recommendations and implementation of those new recommendations

```
EXEC DBMS_AUTO_PARTITION.CONFIGURE('AUTO_PARTITION_MODE','OFF');
```

> **Note:**
>
> This mode does not disable existing automatically partitioned tables.

Manage schemas and tables for automatic partitioning

Use the `AUTO_PARTITION_SCHEMA` and `AUTO_PARTITION_TABLE` settings to specify schemas and tables considered for automatic partitioning.

> **Note:**
>
> When automatic partitioning is invoked, all schemas and tables in user-managed schemas are considered for automatic partitioning if both the inclusion and exclusion lists are empty.

- Assuming the inclusion list and the exclusion list are empty, add the `HR` schema and the `SH.SALES` table to the exclusion list, preventing only those objects from automatic partitioning analysis.

```
BEGIN
    DBMS_AUTO_PARTITION.CONFIGURE(
         PARAMETER_NAME  => 'AUTO_PARTITION_SCHEMA',
         PARAMETER_VALUE => 'HR',
         ALLOW           => FALSE);

    DBMS_AUTO_PARTITION.CONFIGURE(
         PARAMETER_NAME  => 'AUTO_PARTITION_TABLE',
         PARAMETER_VALUE => 'SH.SALES',
         ALLOW           => FALSE);
END;
/
```

- After the previous example runs, use the following to remove the `HR` schema from the exclusion list.

```
BEGIN
    DBMS_AUTO_PARTITION.CONFIGURE(
         PARAMETER_NAME  => 'AUTO_PARTITION_SCHEMA',
         PARAMETER_VALUE => 'HR',
         ALLOW           => NULL);
```

```
END;
/
```

- Use the following command to remove all schemas from the exclusion list.

```
BEGIN
    DBMS_AUTO_PARTITION.CONFIGURE(
         PARAMETER_NAME  => 'AUTO_PARTITION_SCHEMA',
         PARAMETER_VALUE => NULL,
         ALLOW           => TRUE);
END;
/
```

- Assuming the inclusion and exclusion lists are empty, the following example adds the `HR` schema to the inclusion list. As soon as the inclusion list is no longer empty, only schemas in the inclusion list are considered.

  With this example, only the `HR` schema is a candidate for automatic partitioning.

```
BEGIN
    DBMS_AUTO_PARTITION.CONFIGURE(
         PARAMETER_NAME  => 'AUTO_PARTITION_SCHEMA',
         PARAMETER_VALUE => 'HR',
         ALLOW           => TRUE);
END;
/
```

Manage Automatic Partitioning Report Retention Period

Set the retention period for automatic partitioning reports to 365 days.

```
BEGIN
    DBMS_AUTO_PARTITION.CONFIGURE(
         PARAMETER_NAME  => 'AUTO_PARTITION_REPORT_RETENTION',
         PARAMETER_VALUE => '365');
END;
/
```

See CONFIGURE Procedure for more information.

Use Automatic Partitioning

Describes the flow and general processes for using and managing automatic partitioning in Autonomous Database.

1. Choose the database for automatic partitioning.
   In general, Oracle recommends using automatic partitioning in cloned or manually created databases rather than production databases. The analysis and verification of automatic partitioning using `RECOMMEND_PARTITION_METHOD` is potentially a resource-intensive and long running operation that can add undesirable processing to your database.

   To use a secondary database for automatic partitioning analysis, the database must have information about your workload in an internally managed SQL workload repository.

   a. Use a cloned database for automatic partitioning.

Autonomous Database automatically collects workload information over time in an internally managed SQL workload repository maintained in the SQL Tuning Set (`SYS_AUTO_STS`). If you clone your production database after having run the workload for a while, the clone will have the necessary workload information. You can use automatic partitioning with such clones without any additional actions.

See Clone an Autonomous Database Instance

b. Use other databases for automatic partitioning.
You can run your workload manually to collect the necessary workload information. If you manually run your workload prior to using automatic partitioning, any Autonomous Database that contains your desired schemas and data can be used for automatic partitioning after your workload was run, regardless of whether it is cloned or manually created.

2. Recommend automatic partitioning.
Use `RECOMMEND_PARTITION_METHOD` to analyze your database, specific schemas, or specific tables to identify the optimal partitioning strategy, if any. The recommendation analyzes your workload and schemas verifying performance benefits by running your workload against an internally created auxiliary table. This can be a resource-intensive and long running operation, requiring CPU and IO to create the auxiliary table and verify performance. You will also temporarily need additional space, of 1 - 1.5 times, your largest candidate table.

3. Apply the recommendation.
Any recommendation can be implemented with the `APPLY_RECOMMENDATION` procedure in the database where the recommendation analysis occurred. Alternatively, any recommendation can be extracted from the database used for analysis and applied to any database, such as a production system. The script needed for manual modification is stored in column `MODIFY_TABLE_DDL` in the `DBA_AUTO_PARTITION_RECOMMENDATION` view.

   Oracle recommends applying automatic partitioning to your database at off-peak time. While your tables will be modified to automatically partitioned tables, the conversion adds additional resource requirements to your system, such as additional CPU and IO. Automatic partitioning requires as much as 1.5 times the size of the table to being modified as additional free space, depending on concurrent ongoing DML operations on those tables.

Generate Automatic Partitioning Reports

Generate automatic partitioning reports using the `REPORT_ACTIVITY` and `REPORT_LAST_ACTIVITY` functions of the `DBMS_AUTO_PARTITION` package.

Generate a report, in plain text format, of automatic partitioning operations for a specific period

This example generates a report containing typical information about the automatic partitioning operations for the last 24 hours. The report is generated in plain text format by default.

```
DECLARE
  Report clob := NULL
BEGIN
    Report := DBMS_AUTO_PARTITION.REPORT_ACTIVITY();
END;
/
```

Generate a report, in HTML format, of automatic partitioning operations for MAY 2021

This example generates a report containing basic information about the automatic partitioning operations for the month of MAY 2021. The report is generated in the HTML format, and it includes only a summary of automatic partitioning operations.

```
DECLARE
  Report clob := NULL
BEGIN
    Report := DBMS_AUTO_PARTITION.REPORT_ACTIVITY(
                      ACTIVITY_START => TO_TIMESTAMP('2021-05-01', 'YYYY-MM-
DD'),
                      ACTIVITY_END   => TO_TIMESTAMP('2021-06-01', 'YYYY-MM-
DD'),
                      TYPE           => 'HTML',
                      SECTION        => 'SUMMARY',
                      LEVEL          => 'BASIC' );
END;
/
```

Generate a report, in plain text format, of the last automatic partitioning operation

This example generates a report containing typical information about the last automatic partitioning operation. The report is generated in the plain text format by default.

```
DECLARE
  Report clob := NULL
BEGIN
    Report := DBMS_AUTO_PARTITION.REPORT_LAST_ACTIVITY();
END;
/
```

See REPORT_ACTIVITY Function for more information.

See REPORT_LAST_ACTIVITY Function for more information.

Example Automatic Partitioning Scenarios

Example scenarios for automatic partitioning using the `DBMS_AUTO_PARTITION` API procedures and functions.

Generate a recommendation for a single table and manually apply the recommendation

1. Set `AUTO_PARTITION_MODE` parameter to `REPORT ONLY` to enable an automatic partitioning recommendation to be made and verified. The recommendation is not applied to the table.

   ```
   BEGIN
       DBMS_AUTO_PARTITION.CONFIGURE(
           PARAMETER_NAME  => 'AUTO_PARTITION_MODE',
           PARAMETER_VALUE => 'REPORT ONLY');
   END;
   /
   ```

2. Validate that `TPCH.LINEITEM` table is a candidate for automatic partitioning. This step is optional and recommended when you are selectively targeting single tables.

```
SELECT DBMS_AUTO_PARTITION.VALIDATE_CANDIDATE_TABLE(
           TABLE_OWNER => 'TPCH',
           TABLE_NAME  => 'LINEITEM')
FROM DUAL;
```

   If the table is a valid candidate, when you invoke automatic partitioning for a recommendation analysis it returns as `VALID`. Otherwise, the violation criteria is shown.

   See VALIDATE_CANDIDATE_TABLE Function for a list of criteria for eligible candidate tables.

3. Invoke the `DBMS_AUTO_PARTITION` API to generate a recommendation for the `TPCH.LINEITEM` table.

```
-- DEFINE SQLPLUS BIND VARIABLE FOR RECOMMENDATION ID
VARIABLE RECOMMENDATION_ID VARCHAR2(32);
BEGIN
  :RECOMMENDATION_ID := DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD(
                            TABLE_OWNER => 'TPCH',
                            TABLE_NAME  => 'LINEITEM');
END;
/
```

   The recommendation analysis and verification that you perform with `DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD` can be a resource-intensive and long running operation and might take considerable time. You should perform this step on a database that is not your primary production system. Oracle recommends giving the verification operation sufficient resources by choosing the HIGH service.

4. Check the recommendation. The view `DBA_AUTO_PARTITION_RECOMMENDATIONS` contains the information on the recommendation. In this example, check the recommended partition key and partition method.

```
SELECT PARTITION_METHOD, PARTITION_KEY
  FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
  WHERE RECOMMENDATION_ID = :RECOMMENDATION_ID;
```

   Additionally, query the same view to get the performance analysis report generated for the workload after the table was partitioned according to the recommendation.

```
SELECT REPORT
    FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
    WHERE RECOMMENDATION_ID = :RECOMMENDATION_ID;
```

5. After manual validation of the recommendation, apply the recommendation. If you are applying the recommendation in the database where the recommendation analysis has taken place, apply the recommendation by executing the `APPLY_RECOMMENDATION` procedure.

```
BEGIN
    DBMS_AUTO_PARTITION.APPLY_RECOMMENDATION(
         RECOMMENDATION_ID => :RECOMMENDATION_ID);
```

```
END;
/
```

If you want to apply the recommendation to a different database, such as your production environment, extract the modification DDL. Then, run the extracted modification DDL in your target database. The query to extract the modification DDL is as follows:

```
SELECT MODIFY_TABLE_DDL
    FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
    WHERE RECOMMENDATION_ID = :RECOMMENDATION_ID;
```

Example output of modification DDL:

```
BEGIN
    -- DBMS_AUTO_PARTITION RECOMMENDATION_ID
C3F7A59E085C2F25E05333885A0A55EA
    --   FOR TABLE "TPCH"."LINEITEM"
    --   GENERATED AT 06/04/2021 20:52:29
    DBMS_AUTO_PARTITION.BEGIN_APPLY(EXPECTED_NUMBER_OF_PARTITIONS => 10);

    EXECUTE IMMEDIATE
    'ALTER TABLE "TPCH"."LINEITEM"
    MODIFY PARTITION BYLIST(SYS_OP_INTERVAL_HIGH_BOUND
        ("L_SHIPDATE", INTERVAL ''10'' MONTH, TIMESTAMP ''1992-01-01
00:00:00''))
    AUTOMATIC /* SCORE=23533.11;  */
    (PARTITION P_NULL VALUES(NULL))
    AUTO ONLINE PARALLEL';

    DBMS_AUTO_PARTITION.END_APPLY;
EXCEPTION WHEN OTHERS THEN
    DBMS_AUTO_PARTITION.END_APPLY;
    RAISE;
END;
```

6. Verify that the table was automatically partitioned, query the catalog views.

```
SELECT T.AUTO, T.PARTITIONING_TYPE, C.COLUMN_NAME
    FROM DBA_PART_TABLES T, DBA_PART_KEY_COLUMNS C
    WHERE T.OWNER = 'TPCH' AND T.TABLE_NAME = 'LINEITEM'
        AND T.OWNER = C.OWNER AND T.TABLE_NAME = C.NAME;
```

Use this query to identify when automatic partitioning was applied to a given table.

```
SELECT APPLY_TIMESTAMP_START, APPLY_TIMESTAMP_END
    FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
    WHERE TABLE_OWNER = 'TPCH' AND TABLE_NAME = 'LINEITEM';
```

See CONFIGURE Procedure for information.

See VALIDATE_CANDIDATE_TABLE Function for information.

See RECOMMEND_PARTITION_METHOD Function for information.

See APPLY_RECOMMENDATION Procedure for information.

Generate a recommendation for eligible tables and manually apply the recommendation

1. Set `AUTO_PARTITION_MODE` parameter to `REPORT ONLY` to enable an automatic partitioning recommendation to be made and verified. The recommendation is not applied to existing tables.

```
BEGIN
    DBMS_AUTO_PARTITION.CONFIGURE(
        PARAMETER_NAME  => 'AUTO_PARTITION_MODE',
        PARAMETER_VALUE => 'REPORT ONLY');
END;
/
```

2. Invoke the `DBMS_AUTO_PARTITION` API to generate a recommendation table.

```
-- DEFINE SQLPLUS BIND VARIABLE FOR RECOMMENDATION ID
VARIABLE RECOMMENDATION_ID VARCHAR2(32);
BEGIN
  :RECOMMENDATION_ID := DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD();
END;
/
```

   The recommendation analysis and verification is a resource-intensive and long running operation and might take considerable time. On secondary, non-production databases, Oracle recommends giving the verification sufficient resources by choosing service HIGH.

3. Query the `DBA_AUTO_PARTITION_RECOMMENDATIONS` view to see which tables were analyzed.

```
SELECT TABLE_OWNER, TABLE_NAME, PARTITION_METHOD, PARTITION_KEY
    FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
    WHERE RECOMMENDATION_ID = :RECOMMENDATION_ID
    ORDER BY RECOMMENDATION_SEQ;
```

4. Use this query to drill-down in the report for a specific table that was analyzed in the run, the `TPCH.LINEITEM` table in this example.

```
SELECT REPORT
    FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
    WHERE RECOMMENDATION_ID = :RECOMMENDATION_ID
        AND TABLE_OWNER = 'TPCH'
        AND TABLE_NAME  = 'LINEITEM';
```

5. Apply the recommendation by executing the `APPLY_RECOMMENDATION` procedure.

```
BEGIN
    DBMS_AUTO_PARTITION.APPLY_RECOMMENDATION(
         RECOMMENDATION_ID => :RECOMMENDATION_ID);
END;
/
```

Alternately, apply recommendations for a specific table that was analyzed, the `TPCH.LINEITEM` table in this example.

```
BEGIN
    DBMS_AUTO_PARTITION.APPLY_RECOMMENDATION(
        RECOMMENDATION_ID => :RECOMMENDATION_ID,
        TABLE_OWNER       => 'TPCH',
        TABLE_NAME        => 'LINEITEM');
END;
/
```

See CONFIGURE Procedure for information.

See RECOMMEND_PARTITION_METHOD Function for information.

See APPLY_RECOMMENDATION Procedure for information.

> **Note:**
>
> Recommendations of automatic partitioning generated by the `RECOMMEND_PARTITION_METHOD` function have a time limit, specified by the `TIME_LIMIT` parameter, with a default of 1 day. If you are analyzing a large system with many candidate tables, a single invocation may not generate a recommendation for all tables that can be partitioned. You can safely invoke the recommendation for auto partitioning repeatedly to generate recommendations for additional tables. When the function is invoked and zero rows are in `DBA_AUTO_PARTITION_RECOMMENDATIONS` for the `RECOMMENDATION_ID`, then the function did not find any additional candidate tables for automatic partitioning.

Generate and automatically apply recommendations for eligible tables

1. Set `AUTO_PARTITION_MODE` parameter to `REPORT ONLY` to enable an automatic partitioning recommendation to be made and verified. The recommendation is not applied to existing tables.

   ```
   BEGIN
       DBMS_AUTO_PARTITION.CONFIGURE(
           PARAMETER_NAME  => 'AUTO_PARTITION_MODE',
           PARAMETER_VALUE => 'IMPLEMENT');
   END;
   /
   ```

2. Invoke the `DBMS_AUTO_PARTITION` API to generate a recommendation table.

   ```
   -- DEFINE SQLPLUS BIND VARIABLE FOR RECOMMENDATION ID
   VARIABLE RECOMMENDATION_ID VARCHAR2(32);
   BEGIN
     :RECOMMENDATION_ID := DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD();
   END;
   /
   ```

   The recommendation analysis and verification is a resource-intensive and long running operation and might take considerable time. On secondary, non-production databases, Oracle recommends giving the verification sufficient resources by choosing service HIGH.

3. Query the DBA_AUTO_PARTITION_RECOMMENDATIONS view to see which tables were analyzed.

```
SELECT TABLE_OWNER, TABLE_NAME, PARTITION_METHOD, PARTITION_KEY
    FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
    WHERE RECOMMENDATION_ID = :RECOMMENDATION_ID
    ORDER BY RECOMMENDATION_SEQ;
```

4. Use the REPORT_LAST_ACTIVITY function to retrieve the report on the actions taken during the last run.

```
SELECT DBMS_AUTO_PARTITION.REPORT_LAST_ACTIVITY() FROM DUAL;
```

See CONFIGURE Procedure for information.

See RECOMMEND_PARTITION_METHOD Function for information.

See REPORT_LAST_ACTIVITY Function for information.

Data Dictionary Views for Automatic Partitioning

There are two new views and one updated view in the data dictionary for information about the automatic partitioning configuration and recommendations in your database.

- DBMS_AUTO_PARTITION DBA_AUTO_PARTITION_CONFIG View
  Displays the current configuration parameter settings for automatic partitioning.
- DBMS_AUTO_PARTITION DBA_AUTO_PARTITION_RECOMMENDATIONS View
  When you run CONFIGURE or RECOMMEND_PARTITION, the results from those procedures is stored in this view. The RECOMMENDATION_ID is used in several procedures and functions.
- DBMS_AUTO_PARTITION Updates to Existing Views
  Discusses the changes to existing views as a result of the implementation of automatic partitioning.

DBMS_AUTO_PARTITION DBA_AUTO_PARTITION_CONFIG View

Displays the current configuration parameter settings for automatic partitioning.

Column	Description
PARAMETER_NAME	Name of the configuration parameter
PARAMETER_VALUE	Value of the configuration parameter
LAST_MODIFIED	Time, in UTC, at which the parameter value was last modified.
MODIFIED_BY	User who last modified the parameter value

DBMS_AUTO_PARTITION DBA_AUTO_PARTITION_RECOMMENDATIONS View

When you run CONFIGURE or RECOMMEND_PARTITION, the results from those procedures is stored in this view. The RECOMMENDATION_ID is used in several procedures and functions.

Column	Description
TABLE_OWNER	Owner of the table
TABLE_NAME	Owner of the table

Column	Description
PARTITION_METHOD	Recommended partition method. See CONFIGURE Procedure
PARTITION_KEY	Recommended partition key. NULL means that analysis complete and recommendation is not to partition the table.
GENERATE_TIMESTAMP	Time, in UTC, when this recommendation was generated.
RECOMMENDATION_ID	ID used with DBMS_AUTO_PARTITION APIs to get additional information about this recommendation.
RECOMMENDATION_SEQ	When a recommendation ID has recommendations for multiple tables, provides the order in which the recommendations were generated. Performance reports are generated assuming that earlier recommendations have been applied. For example, the report for RECOMMENDATION_SEQ = 2 assumes recommendations have been applied for both RECOMMENDATION_SEQ = 1 and RECOMMENDATION_SEQ = 2.
MODIFY_TABLE_DDL	DDL that would be, or was, used to apply the recommendation.
APPLY_TIMESTAMP_START	Time, in UTC, when application of this recommendation was started. NULL if recommendation was not applied.
APPLY_TIMESTAMP_END	Time, in UTC, when application of this recommendation was finished. NULL if recommendation was not applied or if application has not finished.
REPORT	SQL Performance Analyzer report from SQL execution on database after recommendation is applied.

DBMS_AUTO_PARTITION Updates to Existing Views

Discusses the changes to existing views as a result of the implementation of automatic partitioning.

In *_PART_TABLES, the AUTO column (VARCHAR2(3)) was added. Its values are as follows:

- YES - If the table was partitioned by DBMS_AUTO_PARTITION.
- NO - If the table was not partitioned by DBMS_AUTO_PARTITION.

Using Oracle Graph with Autonomous Database

Oracle Graph with Autonomous Database enables you to create graphs from data in your Autonomous Database. With graphs you can analyze your data based on connections and relationships between data entities.

As an Analyst or a Developer you can use graph algorithms and graph pattern queries for ranking, clustering, and path analysis in a graph model of your data. You can use graph features to detect anomalous patterns, identify communities, and find new connections in your data. Then, you can use graphs in your applications, for example, for fraud detection in banking, improved traceability in smart manufacturing, building linked data applications, and more; all while gaining enterprise-grade security, ease of data ingestion, and support for a wide range of workloads.

Autonomous Database includes Graph Studio, which automates graph data management and simplifies modeling, analysis, and visualization across the graph analytics lifecycle. Autonomous Database includes all the graph capabilities from Oracle Database, except the following:

- When using RDF graphs, querying external data sources using SPARQL federated queries is not supported.
- Triple level security in RDF graphs using Oracle Label Security are not supported. See Fine-Grained Access Control for RDF Data for more information.

Database roles must be set to use Graph Studio on an Autonomous Database instance. See Required Roles to Access Tools from Database Actions for more information.

- About Oracle Graph Studio on Autonomous Database
- Building Applications Using Graph APIs with Autonomous Database
  If you are developing an application using graphs, you can also access Property Graph APIs and RDF Graph APIs with Autonomous Database without Graph Studio.

About Oracle Graph Studio on Autonomous Database

Graph Studio allows you to work with the two popular graph models, property graphs and RDF graphs.

Graph Studio features include automated modeling to create graphs from database tables, an integrated notebook to run graph queries and analytics, and native graph and other visualizations. You can easily create and manage either of the graph models. You can validate the graphs by executing queries and exploring their properties.

See Graph Studio: Interactive, Self-Service User Interface and Access Graph Studio Using Oracle Cloud Infrastructure Console for more information on Graph Studio.

Notes for Graph Studio on Autonomous Database:

- Graph Studio is supported with Data Warehouse and Transaction Processing workloads.

Building Applications Using Graph APIs with Autonomous Database

If you are developing an application using graphs, you can also access Property Graph APIs and RDF Graph APIs with Autonomous Database without Graph Studio.

- **Property Graph**: Create and work with Property Graphs with Python and Java developer APIs and the Property Graph Query Language (PGQL). You can use developer APIs to create a graph, run graph queries, and execute graph algorithms.

  You can use these APIs by deploying Oracle Graph Server and Client from OCI Marketplace.

  See Oracle Database Graph for more information.
- **RDF Graph**: Using RDF Graph Server and Query UI you can manage, store, query, and perform inferencing on RDF graphs in Autonomous Database. You can use a REST API, a SPARQL endpoint, a web UI to run SPARQL queries, and developer APIs for advanced RDF graph data management operations.

  You can use these APIs by deploying Oracle RDF Graph Server and Query UI from OCI Marketplace.

  See Using RDF Graph Server and Query UI for more information.

Use Oracle Machine Learning with Autonomous Database

Oracle Machine Learning components are integrated into Autonomous Database.

One of the Oracle Machine Learning Oracle Machine Learning Notebooks is a notebook style application designed for advanced SQL users. Oracle Machine Learning Notebooks provides interactive data analysis that lets you develop, document, share, and automate reports based on sophisticated analytics and data models.
Key features of Oracle Machine Learning Notebooks:

- Allows collaboration among data scientists, developers, business users
- Leverages the scalability and performance of Oracle Platform and its Cloud Services

To use Oracle Machine Learning Notebooks with Autonomous Database you need to add or create Oracle Machine Learning users and then access Oracle Machine Learning notebooks:

- **OML User Management** lets the **Admin** (user with administrative privileges) create and modify Oracle Machine Learning user accounts. See Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database for details on accessing **OML User Management**.
- **OML Application** – application **users** access Oracle Machine Learning Notebooks to create, view, and share notebooks for data analytics, data visualization, and other Oracle Machine Learning Notebooks tasks. Users access Oracle Machine Learning Notebooks by clicking the Oracle Machine Learning Home icon on the User Administration page or from the account details mailed to a new user. See Work with Oracle Machine Learning User Interface for Data Access, Analysis, and Discovery for details on using the OML application.

See Machine Learning with Autonomous Database for more information.

- Machine Learning with Autonomous Database
  Oracle Machine Learning enables you to solve key enterprise business problems and accelerates the development and deployment of data science and machine learning-based solutions using scalable, automated, and secure machine learning for model building, evaluation, and deployment.

Machine Learning with Autonomous Database

Oracle Machine Learning enables you to solve key enterprise business problems and accelerates the development and deployment of data science and machine learning-based solutions using scalable, automated, and secure machine learning for model building, evaluation, and deployment.

Certain Oracle Machine Learning products are available on specific Oracle Database platforms.

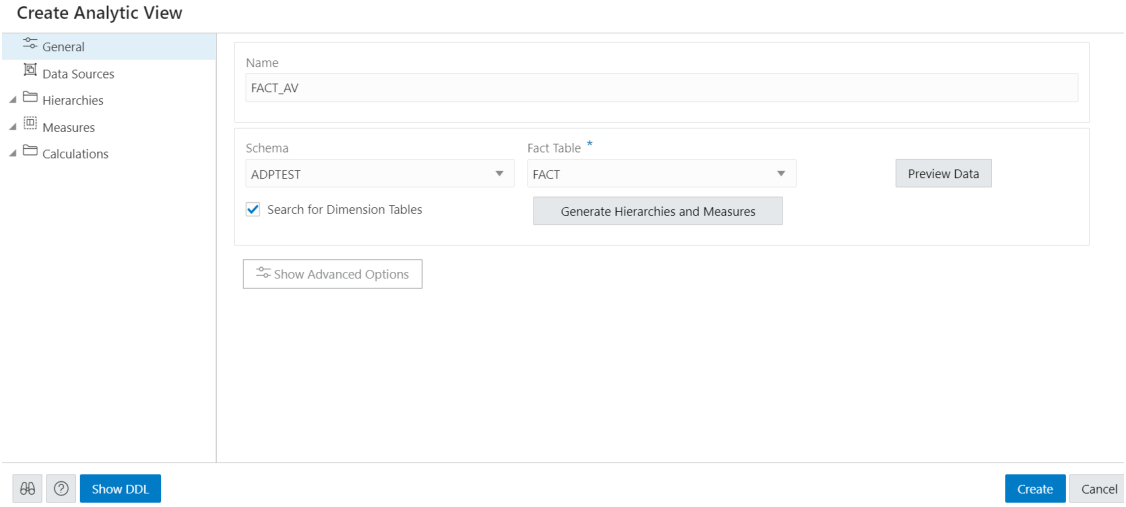
Product Name	Description	Available on:
OML Notebooks	Oracle Machine Learning Notebooks is a collaborative user interface for data scientists, business/data analysts, and SQL, R, and Python users who explore, prepare, and transform data, and who perform machine learning in Oracle Autonomous Database.	Oracle Autonomous Database

Product Name	Description	Available on:
OML AutoML User Interface	AutoML User Interface (AutoML UI) is an Oracle Machine Learning interface that provides you no-code automated machine learning. Business users without extensive data science background can use AutoML UI to create and deploy machine learning models, while data scientists can use it as a productivity tool for quickly assessing algorithms and hyperparameters.	Oracle Autonomous Database
OML for SQL	Oracle Machine Learning for SQL (OML4SQL) provides scalable in-database machine learning algorithms for model building and data scoring.	Oracle Database (on premises and Database Cloud Service) and Oracle Autonomous Database
OML for Python	Oracle Machine Learning for Python (OML4Py) is a Python API for performing statistical and machine learning analysis on data in your Oracle Database. It provides a Python interface to the in-database algorithms and supports deployment of user-defined Python functions with optimal system-supported data-parallel and task-parallel invocation. OML4Py also provides Python functions supporting automated machine learning (AutoML).	Oracle Database and Oracle Database Cloud Service 19c and 21c, and Oracle Autonomous Database
OML for R	Oracle Machine Learning for R (OML4R) is an R API for performing statistical and machine learning analysis on data in your Oracle database. It provides an R interface to the in-database algorithms and supports deployment of user-defined R functions with optional system-supported data-parallel and task-parallel invocation.	Oracle Database and Oracle Database Cloud Service, and Oracle Autonomous Database
OML Services	The REST API for Oracle Machine Learning Services provides a REST API supporting model management and deployment, data and model monitoring, and cognitive text capabilities. It supports lightweight scoring for real-time and streaming use cases for both in-database models and ONNX-format models.	Oracle Autonomous Database

Product Name	Description	Available on:
Oracle Data Miner	An extension to Oracle SQL Developer that enables data scientists and citizen data scientists to explore and prepare data, easily build and compare multiple machine learning models, make predictions, and accelerate model deployment.	Oracle Database and Oracle Autonomous Database

Creating Analytic Views

You can create Analytic Views and view information about them. You can also edit and perform other actions on them.



When you create an Analytic View, you identify a fact table that contains the data to inspect. The **Generate Hierarchies and Measures** button looks at the contents of that table, identifies any hierarchies in the fact table, and searches for other tables that may contain related hierarchies.

While creating an Analytic View, you can enable or disable the following advanced options:

- Autonomous Aggregate Cache, which uses the dimensional metadata of the Analytic View to manage a cache and that improves query response time.
- Analytic View Transparency Views, which presents Analytic Views as regular database views and enables you to use your analytic tools of choice while gaining the benefits of Analytic Views.
- Analytic View Base Table Query Transformation, which enables you to use your existing reports and tools without requiring changes to them.

Create Analytic View

To create Analytic View, click **Create** from the Data Analysis home page and select **Create Analytic View** to begin the process.

Click **Cancel** to cancel the creation of the Analytic View at any time.

Specify Attributes of the Analytic View

On the **General** tab of the **Create Analytic View** pane, specify the following:

- The name for the Analytic View
- The fact table for the view
- Advanced options

You can also preview the data of the fact table and see statistics about that data.

In the **Name** field, specify a name of your choice.

The **Schema** field has the current user's schema. You can only create an Analytic View in that schema.

In the **Fact Table** field, expand the drop-down list and click **More Sources**. The **Select Sources** dialog box has a list of the available tables and views. Select a table or view from the list.

To filter the list, begin typing characters in the Filter field. As you type, the list changes to show the tables or views that contain the characters. Clear the field to show the complete list again. After you select a table or view, click OK.

To enable or disable the advanced options, on the **Create Analytic View** pane, click the **Show Advanced Options** icon at the bottom left. Select or deselect options as desired.

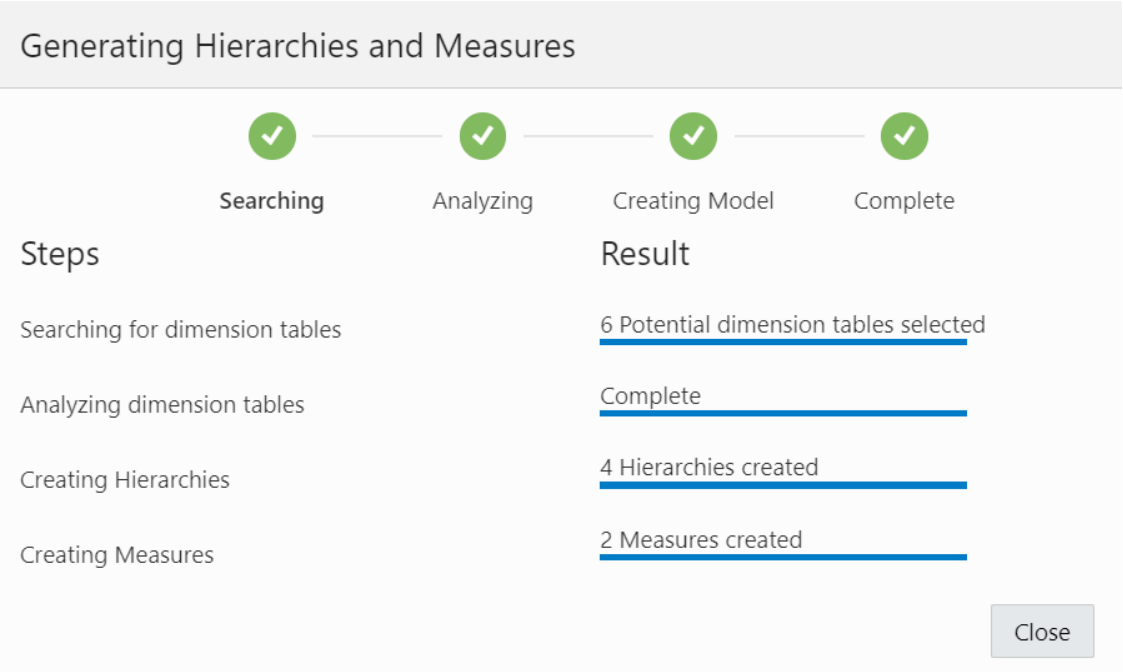
To view the data in the fact table and statistics about the data, click the **Preview Data** button. In the Preview and Statistics pane, the Preview tab shows the columns of the table and the data in the columns.

The Statistics tab shows the size of the table and the number of rows and columns. The statistics may take a few moments to appear, during which time the message, "No statistics available..." may appear. The statistics include the names of the columns, their data types, the number of distinct values and null values, the maximum and minimum values, and other information. The bar graph displays the top unique column values and the number of their occurrences for the selected column. Point to a bar in the graph to see the number of occurrences of the unique value.

Click **Close** to close the Preview and Statistics pane and return to the Create Analytic View pane.

Click on **Generate Hierarchies and Measures** icon.

The Generating Hierarchies and Measures dialog box displays the progress of searching for dimension tables, analyzing the dimension tables and identifying and creating the data sources, joins, hierarchies, and measures to use. When the process completes, click **Close**.



The **Search for Dimension Tables** check box when selected, enables you to search for dimension tables while generating hierarchies and measures.

After the hierarchies and measures are generated, they are displayed under their respective tabs. Review the hierarchies and measures created for you.

Specify the **Name**, **Fact Table** and select **Advanced Options** in the **General** tab of **Create Analytic View** pane. Click **Create** to generate an Analytic View.

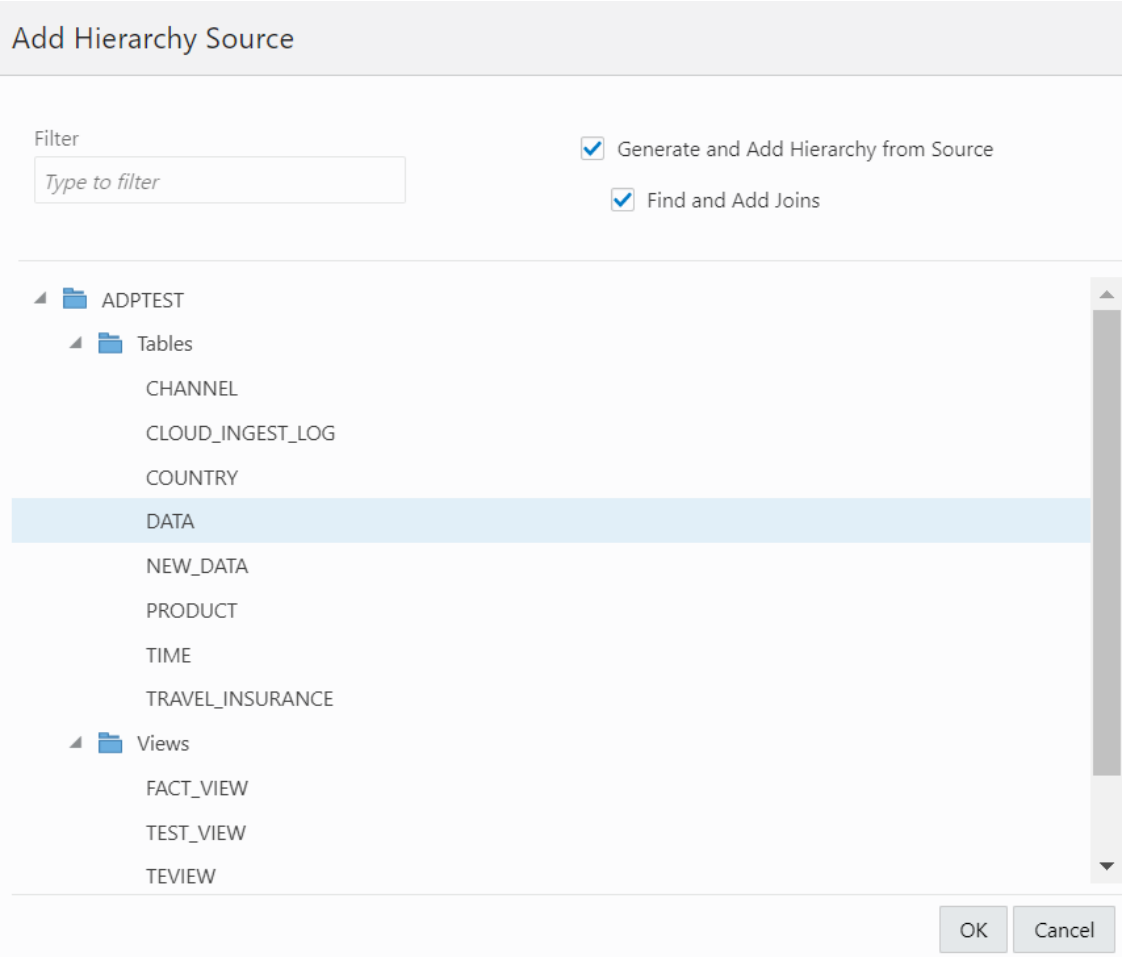
View Data Sources

The Data Sources tab displays the sources of the data and the relationships among them. It has a graphical display of the fact table and the related dimension tables. For example, a fact table of health insurance data might have columns for geography identifiers, income codes, and gender codes. The Data Sources tab would display items for the fact table and for the geography, income, and gender dimension tables.

You can add hierarchies from data sources even after generating hierarchies from the existing fact table. You can add one or more hierarchies to your new or existing analytic view. Multiple hierarchies can be defined and used in an analytical view, however only one will be used by default.

Right-click the Data Sources tab and select **Add Hierarchy Sources** or select **Add Hierarchy Sources**.

Selecting **Add Hierarchy Sources** launches an **Add Hierarchy Source** dialog box.



You can view all the fact tables and views associated with the analytic view.

In the filter field, you can either manually look for the source or start typing to search for the fact table or views from the list of available fact tables and views. After typing the full name of the source, the tool automatically matches the fact table or view.

Select **Generate and Add hierarchy from Source** to generate analysis and hierarchies associated with the source data you select.

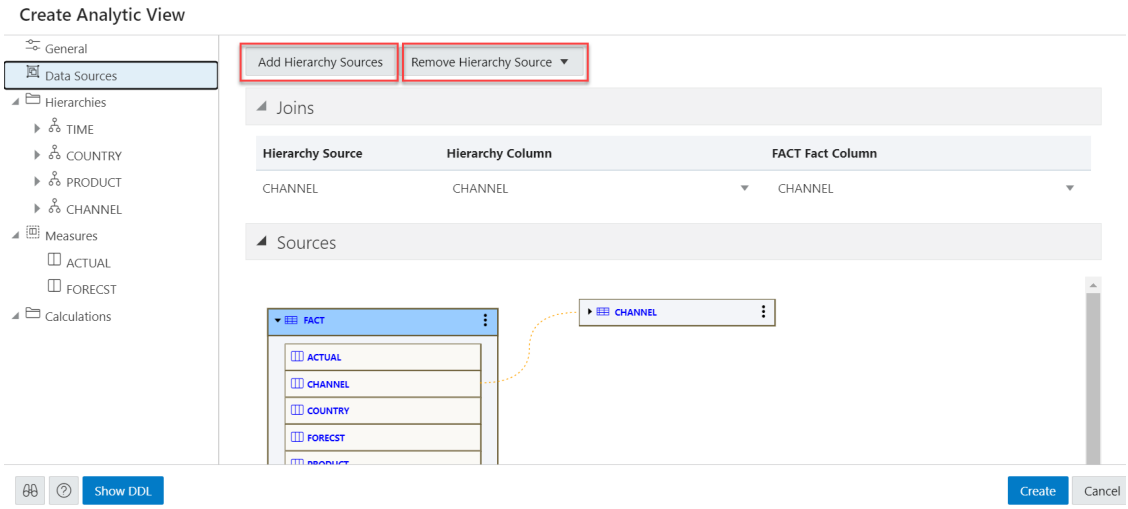
Select **Find and Add Joins** to link all the data sources with the fact table. You can add multiple join entries for a single hierarchy.

Click **OK** to select the source.

The Generating Hierarchies and Measures dialog box displays the progress of analyzing the dimension tables and creating the hierarchies. When the process completes, click **Close**.

> **Note:**
>
> When you add a hierarchy from the data source, you see the new hierarchy in the list of hierarchies in the Hierarchies tab. You can navigate between the Data Sources tab, the Hierarchies tab, the Measures tab, the Calculations tab. You can add a hierarchy from a source that is not connected by navigating back to the Data Sources tab.

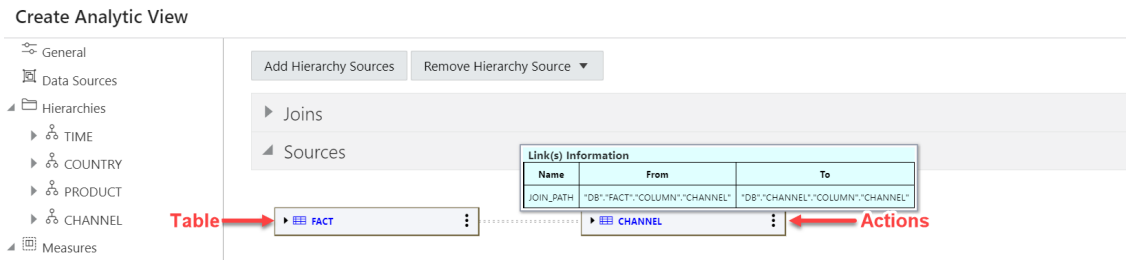
Select **Remove Hierarchy Source** to remove the hierarchies you create from the data sources. You cannot remove hierarchies generated from the fact table you select from this option.



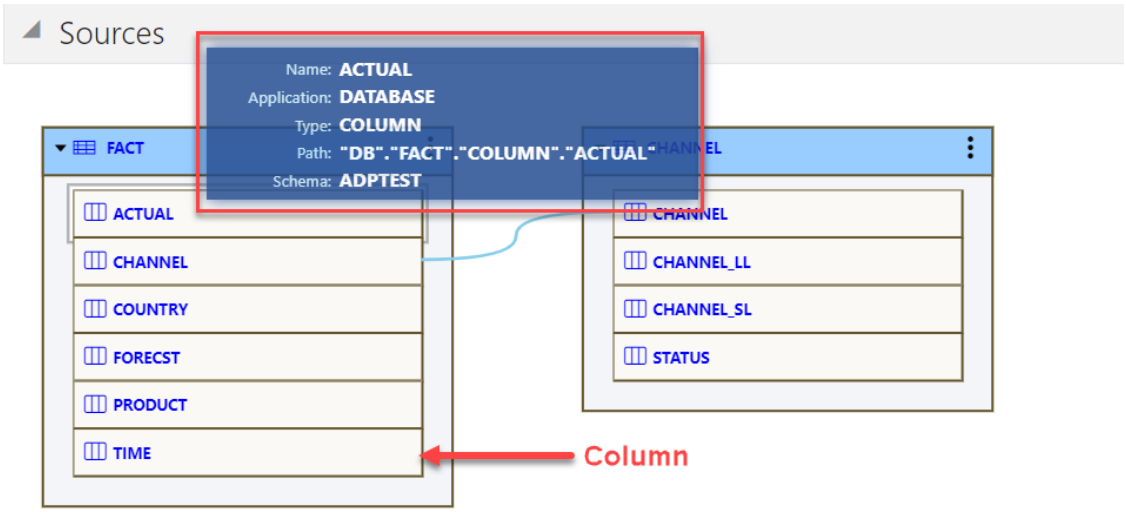
Expand **Joins** to view the **Hierarchy Source**, **Hierarchy Column** and the **Fact column** mapped with the Analytic View. The **Joins** is visible only when the hierarchy table differs from the fact table. You can add multiple join entries for a single hierarchy.

Expand **Sources** to view the fact table associated with the Analytic View. The data model expands to include the data from the source that you added.

Pointing to an item displays the name, application, type, path and the schema of the table. Click the **Actions** (three vertical dots) icon at the right of the item to display a menu to expand or collapse the view of the table.



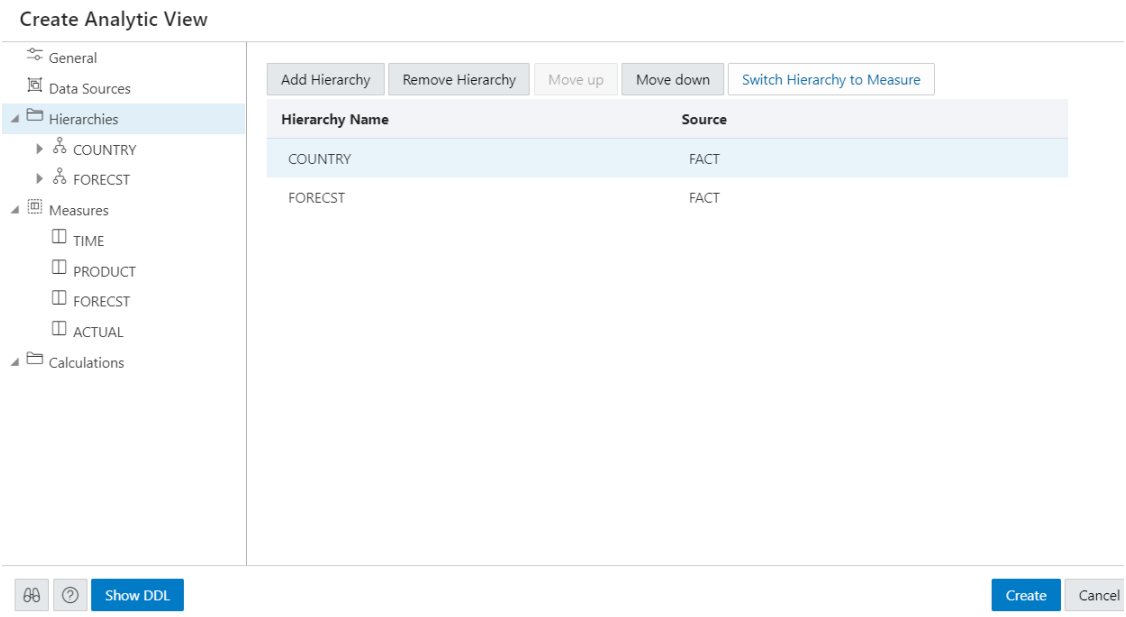
An expanded item displays the columns of the table. Pointing to a column displays the name, application, type, path, and schema of the column.



The lines that connect the dimension tables to the fact table indicate the join paths between them. Pointing to a line displays information about the join paths of the links between the tables. If the line connects a table that is collapsed, then the line is dotted. If the line connects two expanded tables, then the line is solid and connects the column in the dimension table to the column in the fact table.

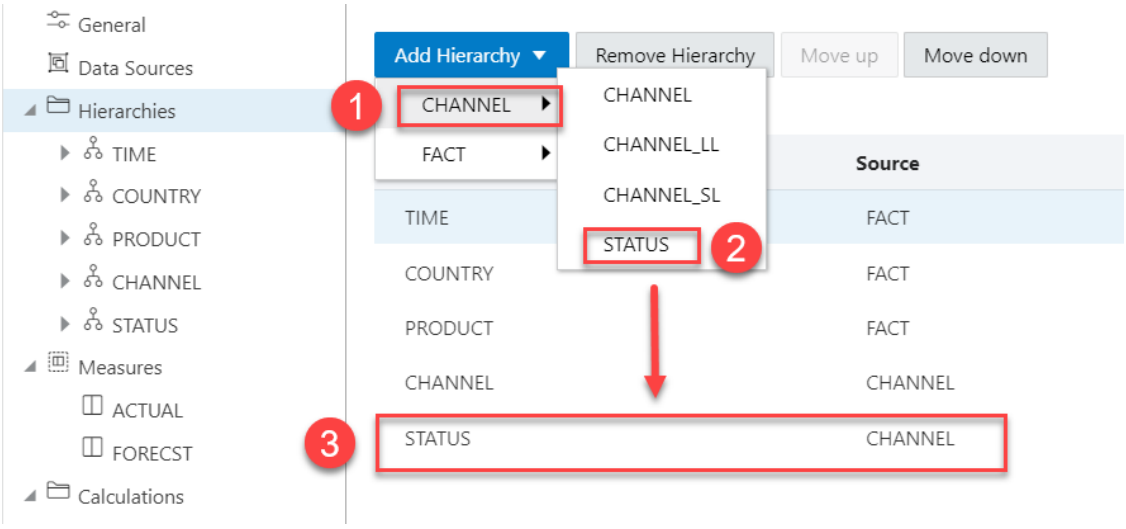
View and Manage Hierarchies

The Hierarchies tab displays the hierarchies generated by the Analytic View creation tool. The display includes the name of the hierarchy and the source table.



An analytic view must include at least one hierarchy.

To add a Hierarchy, click **Add Hierarchy**. This results in a display as a list of column in that table. Select a column that operates as the detailed level of the hierarchy and be the join-key to the fact table.



To remove the hierarchy, select the hierarchy you want to remove from the list and click **Remove Hierarchy**

Select **Move Up** or **Move Down** to position the order of the Hierarchy in the resulting view.

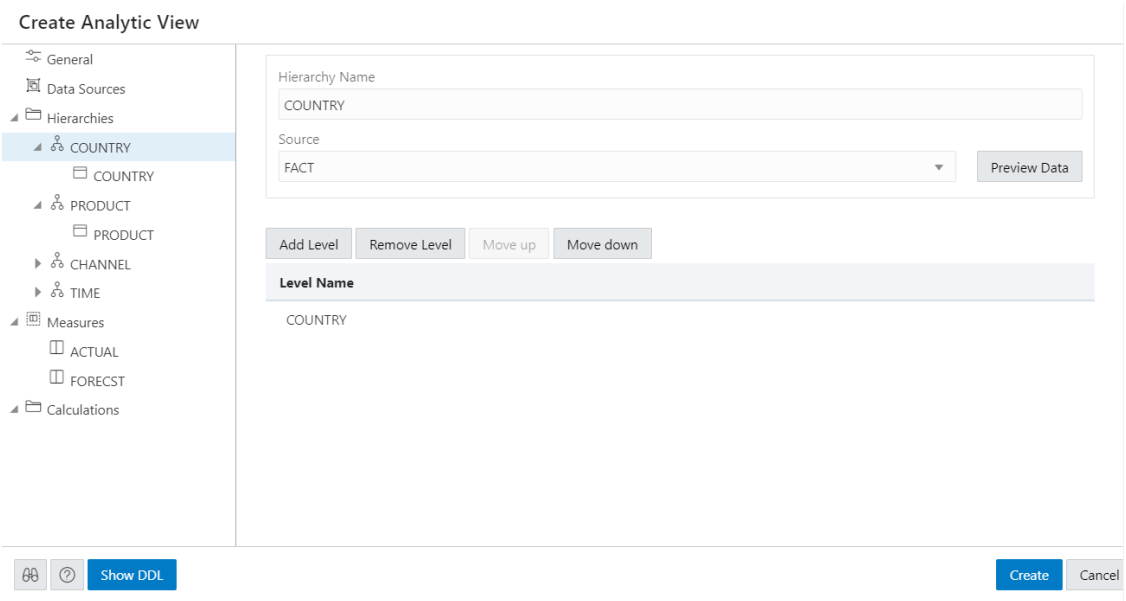
Click **Switch Hierarchy to Measure** to change the hierarchy you select to a measure in the Measures list.

You can also **Add Hierarchy** and **Add Hierarchy From Table** by right-clicking the Hierarchy tab.

If you click on a hierarchy name, a dialog box displays the Hierarchy Name and Source.

To change the source, select a different source from the drop-down list.

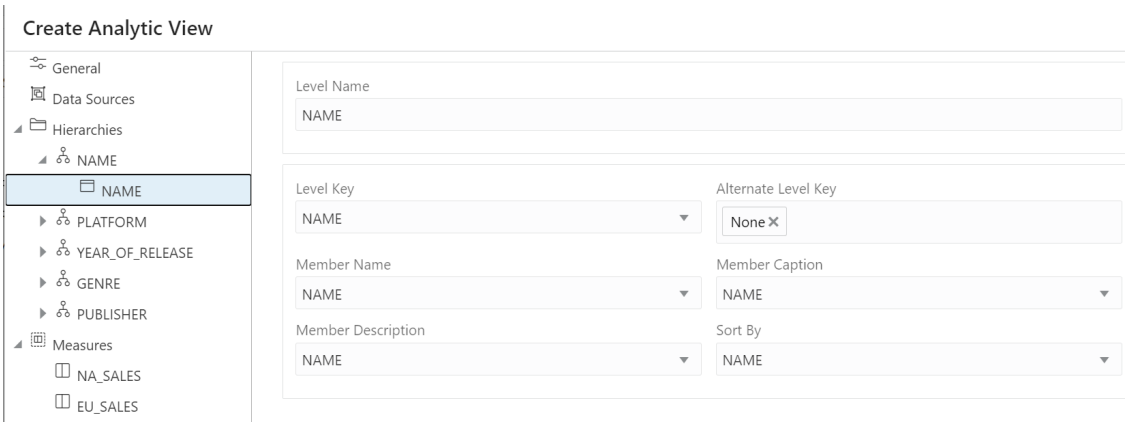
Select **Add Level** to add a level to the hierarchy. Click **Remove Level** to remove the selected level from the hierarchy.



To view the data in the fact table and statistics about the data, click the **Preview Data** button. In the Preview and Statistics pane, the Preview tab shows the columns of the table and the data in the columns. The Statistics tab shows the size of the table and the number of rows and columns.

If you click on a particular level in the Hierarchy tab, a dialog box displays it's respective Level Name, Level Key, Alternate Level Key, Member Name, Member Caption, Member Description, source, and Sort By drop-down. To change any of the field values, enter the value in the appropriate field.

> **Note:**
>
> You can enter multiple level keys Member Name, Member Caption, Member Description and Sort By.



Member Captions and Member Descriptions generally represent detailed labels for objects. These are typically end-user-friendly names. For example, you can caption a hierarchy representing geography areas named GEOGRAPHY_HIERARCHY as "Geography" and specify its description as "Geographic areas such as cities, states, and countries."

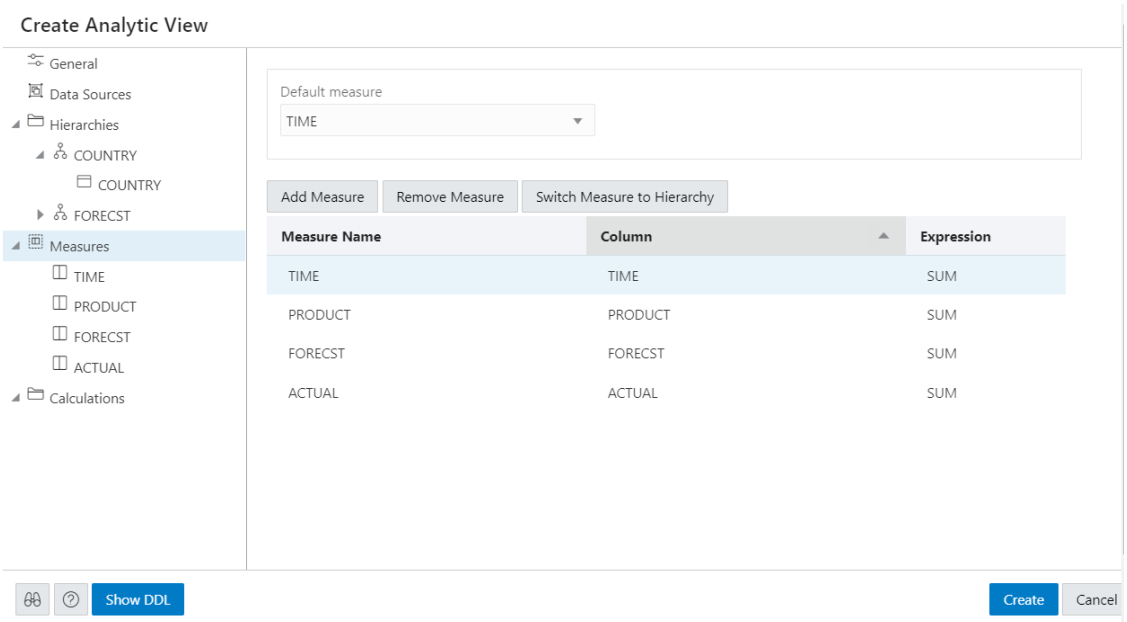
To see the measures for the Analytic View, click **Measures** tab. To immediately create the Analytic View, click **Create**. To cancel the creation, click **Cancel**.

View and Manage Measures

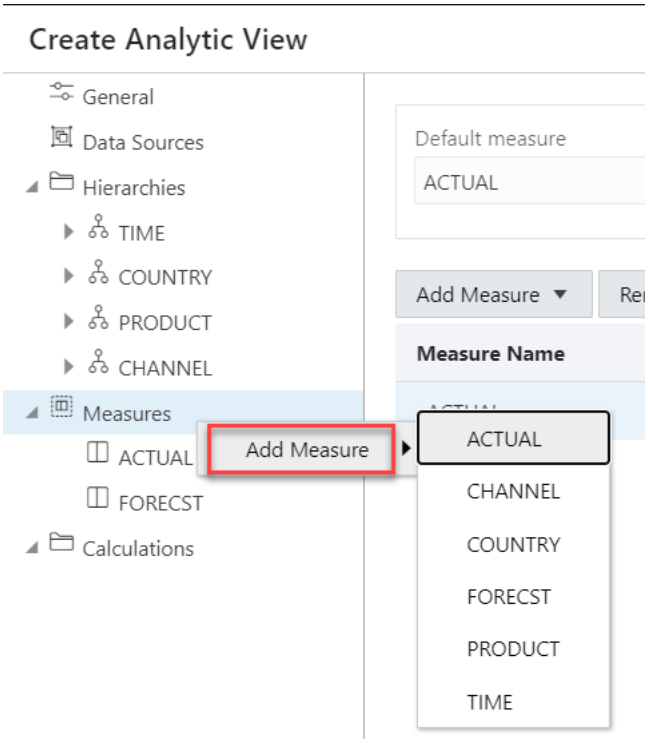
The Measures tab displays the measures suggested for the Analytic View. It displays the Measure Name, Column, and operator Expression for each measure.

The measures specify fact data and the calculations or other operations to perform on the data.

To add measures, click **Add Measure**. You can view a new measure at the bottom of the measures list. To remove the measure, select the measure you want to remove from the list and click **Remove Measure**.



To alternatively add a measure from the data source, right- click the Measures tab. This pops up a list of columns that can be used as measures. Select one measure from the list.



You can exclude a column from the measures on right-clicking the Measures tab and selecting Remove Measure.

Click **Switch Measure to Hierarchy** to change the measure you select to hierarchy in the Hierarchies list.

You must specify a measure as the default measure for the analytic view; otherwise, the first measure in the definition is the default. Select **Default Measure** from the drop-down.

To add a measure, right-click the Measures tab and select **Add Measure**. To remove a measure, select the particular measure you want to remove, right-click on it and select **Remove Measure**.

You can select a different column for a measure from the Column drop-down list. You can select a different operator from the Expression drop-down list.

In creating an analytic view, you must specify one or more hierarchies and a fact table that has at least one measure column and a column to join to each of the dimension tables outside of the fact table.

Create new calculated measures

You can add measure calculations to a query of an analytic view.

The measures and hierarchies associated with the analytic views enable us to create new calculated measures.

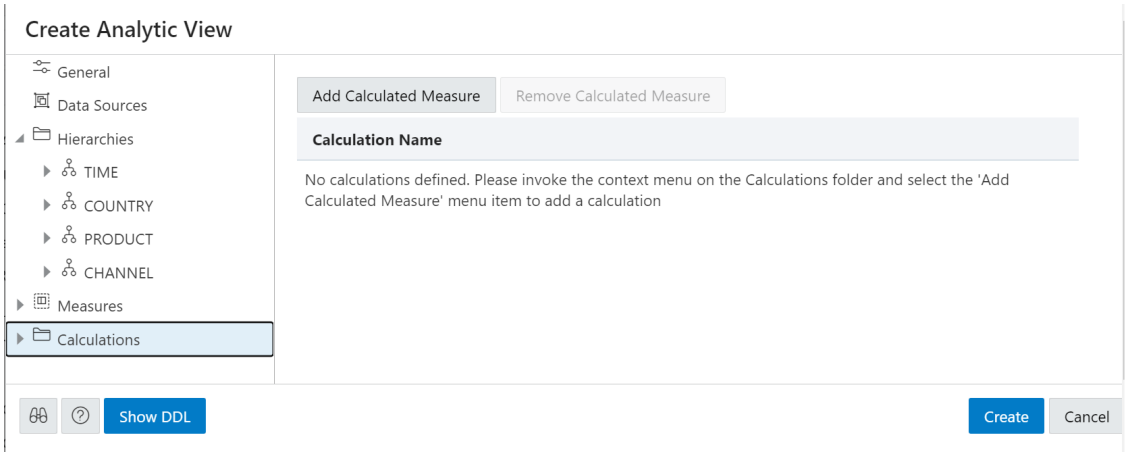
Calculated measures return values from data stored in one or more measures. You compute these measures at run time.

> **Note:**
>
> You can create the measures without increasing the size of the database since the calculated measures do not store the data. However, they may slow performance. You need to decide which measures to calculate on demand.

The Analytic Views provides easy-to-use templates for creating calculated measures.

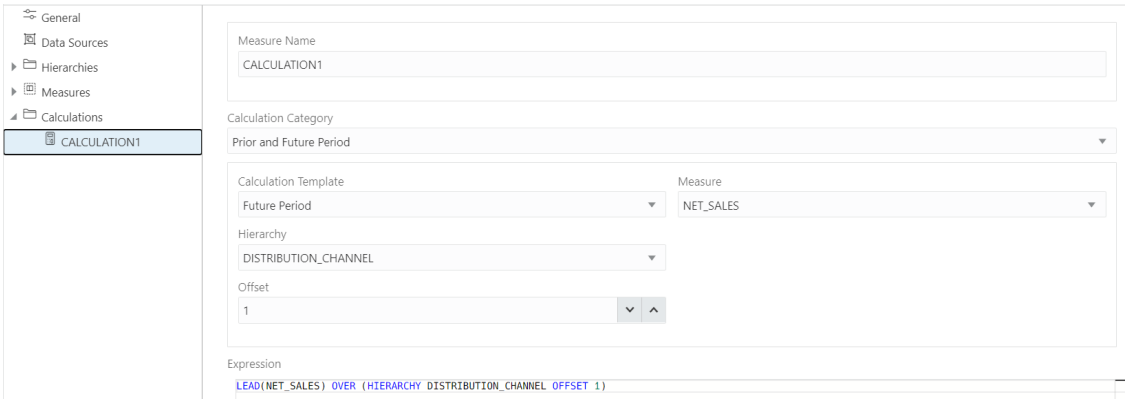
Once you create a calculated measure, it appears in the list of measures of the Analytic View .You can create a calculated measure at any time which is available for querying in SQL.

The Data Analysis tool provides easy-to-use templates for creating calculated measures.



Click **Add Calculated Measure** to add calculations to the measures. You can view the new calculation with system generated name in the **Calculations** tab.

Click the newly created calculated measure.



In the **Measure Name** field, enter the name of the calculated measure.

You can select preferred category of calculation from a list of options such as Prior and Future Period, Cumulative Aggregates, Period To Date, Parallel Period, Moving Aggregates, Share, Qualified Data Reference, and Ranking using the **Calculation Category** drop-down.

Your choice of category of calculation dynamically changes the **Calculation Template**.

For more details on how to use Calculation templates, see Using Calculation Templates.

Select the **Measure** and **Hierarchy** on which you want to base the calculated measures.

Select **Offset** value by clicking the up or the down arrow. The number specifies the number of members to move either forward or backward from the current member. The ordering of members within a level is dependent on the definition of the attribute dimension used by the hierarchy. The default value is 0 which represents POSITION FROM BEGINNING.

The Expression field lists the expressions which the calculated measure uses.

On the creation of the Analytic view, the calculated measure appears in the navigation tree in the Calculated Measures folder.

Click **Create**. A confirmation dialog box appears that asks for your confirmation. Select **Yes** to proceed with the creation of Analytic View.

After creating the Analytic View, you will view a success message informing you of its creation.

On editing the Analytic View you create, you can view the calculated measure in the navigation tree in the Calculations folder.

Click the **Tour** icon for a guided tour of the worksheet highlighting salient features and providing information if you are new to the interface.

Click the **help** icon to open the contextual or online help for the page you are viewing.

Click **Show DDL** to generate Data Definition Language statements for the analytic view.

Edit Analytic View

You might want to edit an Analytic View to make changes to the data sources, the hierarchies, or the measures.

To edit an Analytic View, click the **Action** icon on the Analytic View item, then click **Edit Analytic View**. On the Edit Analytic View screen, select a tab and make changes as desired.

When you have completed the changes, click **Update**.

Use Select AI to Generate SQL from Natural Language Prompts

Oracle Autonomous Database Select AI enables you to query your data using natural language.
The Select AI feature allows Autonomous Database to use generative AI with Large Language Models (LLMs) to convert user's input text into Oracle SQL. Select AI processes the natural language prompt, supplements the prompt with metadata, and then generates and runs a SQL query.

- Usage Guidelines
  Provides usage guidelines that ensure effective and proper usage of natural language prompts for SQL generation to ensure an enhanced user experience.
- About SQL Generation
  Using natural language to interact with your database data is now achievable with LLMs. This means you can use natural language, for example plain English, to query the database.
- Use DBMS_CLOUD_AI to Configure AI Profiles
  Oracle Autonomous Database Serverless uses AI profiles to facilitate and configure access to an LLM and to setup for the generation of SQL statements from natural language prompts.
- Configure DBMS_CLOUD_AI Package
  Describes the steps to use `DBMS_CLOUD_AI`.
- Create and Set an AI Profile
  Describes the steps to create and enable an AI profile.
- Use AI Keyword to Enter Prompts
  Use `AI` as the keyword in a `SELECT` statement for interacting with the database using natural language prompts.
- Examples of Using Select AI
  Explore integrating Oracle's Select AI with various AI providers like OpenAI, Cohere, Azure Open AI, and OCI Generative AI to generate SQL queries directly from natural language.
- Terminology
  Definition of some of the terms used in Select AI feature are described.

Usage Guidelines

Provides usage guidelines that ensure effective and proper usage of natural language prompts for SQL generation to ensure an enhanced user experience.

Intended Use

This feature is intended for the generation and running of SQL queries resulting from user-provided natural language prompts. It automates what a user could do manually based on their schema metadata in combination with a large language model (LLM) of their choice.

While any prompt can be provided, including those that do not relate to the production of SQL query results, Select AI focuses on SQL query generation. Select AI enables submitting general requests with the `chat` action.

Prompt Augmentation Data

The database augments the user-specified prompt with database metadata to mitigate hallucinations from the LLM. The augmented prompt is then sent to the user-specified LLM to produce the query.

The database augments the prompt with schema metadata only. This metadata may include schema definitions, table and column comments, and content available from the data dictionary and catalog. For the purposes of SQL generation, the database does not provide table or view contents (actual row or column values) when augmenting the prompt.

The `narrate` action, however, does provide the result of the query, which may contain database data, to the user-specified LLM from which to generate natural language text describing the query results.

> **WARNING:**
>
> Large language models (LLMs) have been trained on a broad set of text documentation and content, typically from the Internet. As a result, LLMs may have incorporated patterns from invalid or malicious content, including SQL injection. Thus, while LLMs are adept at generating useful and relevant content, they also can generate incorrect and false information including SQL queries that produce inaccurate results and/or compromise security of your data.
>
> The queries generated on your behalf by the user-specified LLM provider will be run in your database. Your use of this feature is solely at your own risk, and, notwithstanding any other terms and conditions related to the services provided by Oracle, constitutes your acceptance of that risk and express exclusion of Oracle's responsibility or liability for any damages resulting from that use.

About SQL Generation

Using natural language to interact with your database data is now achievable with LLMs. This means you can use natural language, for example plain English, to query the database.

When you use Select AI, Autonomous Database manages the process of converting natural language into SQL. This means you can provide a natural language prompt instead of SQL code to interact with your data. Select AI serves as a productivity tool for SQL users and developers and enables non-expert SQL users to derive useful insights from their data, without having to understand data structures or technical languages.

The `DBMS_CLOUD_AI` package in Autonomous Database enables integration with a user-specified LLM for generating SQL code using natural language prompts. The package assists in supplying the LLM with knowledge of the database schema and instructing it to write a SQL query consistent with that schema. The `DBMS_CLOUD_AI` package works with AI providers like OpenAI, Cohere, Azure OpenAI Service, and Oracle Cloud Infrastructure Generative AI.

> **Note:**
>
> Users must have an account with the AI provider and provide their credentials through `DBMS_CLOUD_AI` objects that the Autonomous Database uses.

Use DBMS_CLOUD_AI to Configure AI Profiles

Oracle Autonomous Database Serverless uses AI profiles to facilitate and configure access to an LLM and to setup for the generation of SQL statements from natural language prompts.

AI profiles include database objects that are the target for natural language queries. Metadata used from these targets can include database table names, column names, column data types, and comments. You create and configure AI profiles using the `DBMS_CLOUD_AI.CREATE_PROFILE` and `DBMS_CLOUD_AI.SET_PROFILE` procedures.

In addition to specifying tables and views in the AI profile, you can also specify tables mapped with external tables, including those described in Query External Data with Data Catalog. This enables you to query data not just inside the database, but also data stored in a data lake's object store.

Configure DBMS_CLOUD_AI Package

Describes the steps to use `DBMS_CLOUD_AI`.

The following are required to run `DBMS_CLOUD_AI`:

- Access to an Oracle Cloud Infrastructure cloud account and to an Autonomous Database instance.
- A paid API account for a supported AI provider, one of:
  - OpenAI: See Use OpenAI to get your API keys.
  - Cohere. See Use Cohere to get your secret API keys.
  - Azure OpenAI Service. See Use Azure OpenAI Service for more information on how to configure Azure OpenAI Service.
  - OCI Generative AI. See How to Generate the API Signing Key.
- Network ACL privileges to access your external AI provider.

  > **Note:**
  >
  > Network ACL is not applicable for OCI Generative AI.

- A credential that provides access to the AI provider.

Configure `DBMS_CLOUD_AI`

To configure `DBMS_CLOUD_AI`:

1. Grant the `EXECUTE` privilege on the `DBMS_CLOUD_AI` package to the user who wants to use Select AI.

   By default, only ADMIN user is granted the `EXECUTE` privilege. The ADMIN user can grant `EXECUTE` privilege to other users.

2. Grant network ACL access to the user who wants to use Select AI and for the AI provider endpoint.

   The ADMIN user can grant network ACL access. See APPEND_HOST_ACE Procedure for more information.

3. Create a credential to enable access to your AI provider.
   See CREATE_CREDENTIAL Procedure for more information.

The following example grants the `EXECUTE` privilege to `ADB_USER`:

```
grant execute on DBMS_CLOUD_AI to ADB_USER;
```

The following example grants `ADB_USER` the privilege to use the *api.openai.com* endpoint.

```
BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'api.openai.com',
         ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                             principal_name => 'ADB_USER',
                             principal_type => xs_acl.ptype_db)
   );
END;
/
```

APPEND_HOST_ACE Function Parameters

Parameter	Description
`host`	The host, which can be the name or the IP address of the host. You can use a wildcard to specify a domain or a IP subnet. The host or domain name is not case sensitive. For OpenAI, use *api.openai.com*. For Cohere, use *api.cohere.ai*. For Azure OpenAI Service, use *<azure_resource_name>.openai.azure.com*. See Profile Attributes to know more about `azure_resource_name`.
`ace`	The access control entries (ACE). The `XS$ACE_TYPE` type is provided to construct each ACE entry for the ACL. For more details, see Creating ACLs and ACEs.

Here is an example of how to create a credential to enable access to OpenAI.

```
EXEC DBMS_CLOUD.CREATE_CREDENTIAL('OPENAI_CRED', 'OPENAI', 'your_api_token');
```

DBMS_CLOUD.CREATE_CREDENTIAL Parameters

Parameter	Description
`credential_name`	The name of the credential to be stored. The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
`username`	The `username` and `password` arguments together specify your AI provider credentials. The `username` is a user-specified user name.

Parameter	Description
`password`	The `username` and `password` arguments together specify your AI provider credentials. The `password` is your AI provider secret API key, and depends on the provider: • **OpenAI**: See Use OpenAI to get your API keys. • **Cohere**: See Use Cohere to get your API keys. • **Azure OpenAI Service**: See Use Azure OpenAI Service to get your API keys and how to configure the service. **Note:** If you are using the Azure OpenAI Service principle to authenticate, you can skip the `DBMS_CLOUD.CREATE_CREDENTIAL` procedure. See Examples of Using Select AI for an example of authenticating using Azure OpenAI Service principle.

- Use OpenAI
  To enable OpenAI to generate SQL from natural language prompts, obtain API keys from your OpenAI paid account.
- Use Cohere
  To enable Cohere to generate SQL from natural language prompts, obtain API keys from your Cohere paid account.
- Use Azure OpenAI Service
  To enable Azure OpenAI Service to generate SQL from natural language prompts, configure and provide access to the AI provider.

Use OpenAI

To enable OpenAI to generate SQL from natural language prompts, obtain API keys from your OpenAI paid account.

You can find your secret API key in your User settings.

Use Cohere

To enable Cohere to generate SQL from natural language prompts, obtain API keys from your Cohere paid account.

Click **Dashboard**, and click **API Keys** on the left navigation. Copy the default API key or create another key. See API-Keys for more information.

Use Azure OpenAI Service

To enable Azure OpenAI Service to generate SQL from natural language prompts, configure and provide access to the AI provider.

To use Azure OpenAI Service, perform the following steps:

1. Obtain your secret API keys. You can find your API keys in the Resource Management section of your Azure portal. On your Azure OpenAI Service Resource page, click **Keys and Endpoint**. You can copy either KEY1 or KEY2.

2. Create an Azure OpenAI Service resource and deploy a model: Create and deploy an Azure OpenAI Service resource.

> **Tip:**
>
> - Note the resource name and deployment name as those parameters are used to provide network access permission and creating your Azure OpenAI Service profile using the `DBMS_CLOUD_AI.CREATE_PROFILE` procedure.
> - To know more about rate limits for token per minute on a model, see Azure OpenAI Service quotas and limits.

3. Allow access to Azure OpenAI Service:
   - You can use your secret API key to allow access to Azure OpenAI Service. To know more, see the example in Examples of Using Select AI.
   - Allow service principal to access Azure OpenAI Service and grant the required permissions to the service principal: Use Azure Service Principal to Access Azure Resources and Role-based access control for Azure OpenAI Service. To know more, see the example in Examples of Using Select AI.

Create and Set an AI Profile

Describes the steps to create and enable an AI profile.

Use `DBMS_CLOUD_AI.CREATE_PROFILE` to create an AI profile. Next start `DBMS_CLOUD_AI.SET_PROFILE` to enable the AI profile so that you can use `SELECT AI` with a natural language prompt.

> **Note:**
>
> You must run `DBMS_CLOUD_AI.SET_PROFILE` in each new database session (connection) before you use `SELECT AI`.

The following example with the OpenAI provider creates an `AI` profile called `OPENAI` and sets the `OPENAI` profile for the current user session.

```
-- Create AI profile
--
SQL> BEGIN
  DBMS_CLOUD_AI.create_profile(
      'OPENAI',
      '{"provider": "openai",
        "credential_name": "OPENAI_CRED",
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "sales"},
                        {"owner": "SH", "name": "products"},
                        {"owner": "SH", "name": "countries"}]
       }');
END;
/
```

```
PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
SQL> EXEC DBMS_CLOUD_AI.set_profile('OPENAI');

PL/SQL procedure successfully completed.
```

Use AI Keyword to Enter Prompts

Use `AI` as the keyword in a `SELECT` statement for interacting with the database using natural language prompts.

The `AI` keyword in a `SELECT` statement instructs the SQL execution engine to use the LLM identified in the active AI profile to process natural language and to generate SQL.

You can use the `AI` keyword in a query with Oracle clients such as SQL Developer, OML Notebooks, and third-party tools, to interact with database in natural language.

> **Note:**
>
> You cannot run PL/SQL statements, DDL statements, or DML statements using the `AI` keyword.

Syntax

The syntax for running AI prompt is:

```
SELECT AI action natural_language_prompt
```

Parameters

The following are the parameters available for the *action* parameter:

Parameter	Description
`runsql`	Run the provided SQL command using a natural language prompt. This is the default action and it is optional to specify this parameter.
`showsql`	Displays the SQL statement for a natural language prompt.
`narrate`	The output of the prompt is explained in natural language. This option sends the SQL result to the AI provider to produce a natural language summary.
`chat`	Generates a response directly from the LLM based on the prompt. If `conversation` in the `DBMS_CLOUD_AI.CREATE_PROFILE` function is set to `true`, this option includes content from prior interactions or prompts, potentially including schema metadata.
`explainsql`	The SQL generated from the prompt is explained in natural language. This option sends the generated SQL to the AI provider to produce a natural language explanation.

Usage Notes

- Select `AI` is not supported in Database Actions or APEX Service. You can use only `DBMS_CLOUD_AI.GENERATE` function.
- The `AI` keyword is supported only in a `SELECT` statement.
- You cannot run PL/SQL statements, DDL statements, or DML statements using the `AI` keyword.
- The sequence is `SELECT` followed by `AI`. These keywords are not case-sensitive. After a `DBMS_CLOUD_AI.SET_PROFILE` is configured, the text after `SELECT AI` is a natural language prompt. If an AI profile is not set, `SELECT AI` reports the following error:

  ```
  ORA-00923: FROM keyword not found where expected
  00923. 00000 -  "FROM keyword not found where expected"
  ```

- Special character usage rules apply according to Oracle guidelines. For example, use single quotes twice if you are using an apostrophe in a sentence.

  ```
  select ai how many customers in SF don''t own their own home
  ```

- LLMs are subject to *hallucinations* and results are not always correct:
  - It is possible that `SELECT AI` may not be able to run the generated SQL for a specific natural language prompt.
  - It is possible that `SELECT AI` may not be able to generate SQL for a specific natural language prompt.

  In such a scenario, `SELECT AI` responds with information to assist you in generating valid SQL.
- Use the `chat` action, with `SELECT AI chat`, to learn more about SQL constructs. For better results with the `chat` action, use database views or tables with contextual column names or consider adding column comments explaining values stored in the columns.
- To access DBA or USER views, see DBMS_CLOUD_AI Views.

Examples of Using Select AI

Explore integrating Oracle's Select AI with various AI providers like OpenAI, Cohere, Azure Open AI, and OCI Generative AI to generate SQL queries directly from natural language.

These examples showcase common Select AI actions and guide you through setting up your profile with different AI providers to leverage those actions.

Example: Select AI Actions

The following example illustrate actions such as `runsql`, `showsql`, `narrate`, `chat`, and `explainsql` that you can perform with `SELECT AI`. These examples use the `sh` schema with AI provider and profile attributes set in the `DBMS_CLOUD_AI.CREATE_PROFILE` function.

```
SQL> select ai how many customers exist;

CUSTOMER_COUNT
--------------
         55500
```

```
SQL> select ai showsql how many customers exist;

RESPONSE
--------------------------------------------------
SELECT COUNT(*) AS total_customers
FROM SH.CUSTOMERS


SQL> select ai narrate how many customers exist;

RESPONSE
--------------------------------------------------------
There are a total of 55,500 customers in the database.

SQL> select ai chat how many customers exist;

RESPONSE
--------------------------------------------------------------------------------
--
It is impossible to determine the exact number of customers that exist as it
con
stantly changes due to various factors such as population growth, new
businesses
, and customer turnover. Additionally, the term "customer" can refer to
individu
als, businesses, or organizations, making it difficult to provide a specific
num
ber.


SQL> select ai explainsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS customer_count
FROM SH.CUSTOMERS AS c
WHERE c.CUST_STATE_PROVINCE = 'San Francisco' AND c.CUST_MARITAL_STATUS =
'Married';

Explanation:
- We use the 'SH' table alias for the 'CUSTOMERS' table for better
readability.
- The query uses the 'COUNT(*)' function to count the number of rows that
match the given conditions.
- The 'WHERE' clause is used to filter the results:
  - 'c.CUST_STATE_PROVINCE = 'San Francisco'' filters customers who have 'San
Francisco' as their state or province.
  - 'c.CUST_MARITAL_STATUS = 'Married'' filters customers who have 'Married'
as their marital status.
The result of this query will give you the count of customers in San
Francisco who are married, using the column alias 'customer_count' for the
result.

Remember to adjust the table and column names based on your actual schema if
```

```
they differ from the example.

Feel free to ask if you have more questions related to SQL or database in
general.
```

Example: Select AI with OpenAI

The following example shows how you can use OpenAI to generate SQL statements from natural language prompts.

> **Note:**
>
> Only an ADMIN user can run `EXECUTE` privileges and network ACL procedure.

```
--Grants EXECUTE privilege to ADB_USER
--
SQL> grant execute on DBMS_CLOUD_AI to ADB_USER;

-- Grant Network ACL for OpenAI endpoint
--
SQL> BEGIN
     DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'api.openai.com',
         ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                             principal_name => 'ADB_USER',
                             principal_type => xs_acl.ptype_db)
     );
    END;
    /

PL/SQL procedure successfully completed.

--
-- Create Credential for AI provider
--
SQL> EXEC DBMS_CLOUD.CREATE_CREDENTIAL('OPENAI_CRED', 'OPENAI', '<your api
token>');

PL/SQL procedure successfully completed.

--
-- Create AI profile
--SQL>
BEGIN

DBMS_CLOUD_AI.CREATE_PROFILE(
      'OPENAI',
      '{"provider":
"openai",
        "credential_name": "OPENAI_CRED",
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "countries"},
```

```
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}],
         "conversation": "true"
        }');

END;
     /

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
SQL> EXEC DBMS_CLOUD_AI.SET_PROFILE('OPENAI');

PL/SQL procedure successfully completed.

--
-- Get Profile in current session
--
SQL> SELECT DBMS_CLOUD_AI.get_profile() from dual;

DBMS_CLOUD_AI.GET_PROFILE()
--------------------------------------------------------------------------------
--
"OPENAI"

--
-- Use AI
--
SQL> select ai how many customers exist;

CUSTOMER_COUNT
--------------
         55500

SQL> select ai how many customers in San Francisco are married;

MARRIED_CUSTOMERS
-----------------
               18


SQL> select ai showsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS married_customers_count
FROM SH.CUSTOMERS c
WHERE c.CUST_CITY = 'San Francisco'
  AND c.CUST_MARITAL_STATUS = 'Married'
```

```
SQL> select ai narrate what are the top 3 customers in San Francisco;

RESPONSE
--------------------------------------------------------------------------------
--
The top 3 customers in San Francisco are:

1. Hector Colven - Total amount sold: $52,025.99
2. Milburn Klemm - Total amount sold: $50,842.28
3. Gavin Xie - Total amount sold: $48,677.18


SQL> select ai chat what is Autonomous Database;

RESPONSE
--------------------------------------------------------------------------------
--
Autonomous Database is a cloud-based database service provided by Oracle. It
is
designed to automate many of the routine tasks involved in managing a
database,
such as patching, tuning, and backups. Autonomous Database uses machine
learning
 and automation to optimize performance, security, and availability, allowing
us
ers to focus on their applications and data rather than database
administration
tasks. It offers both Autonomous Transaction Processing (ATP) for
transactional
workloads and Autonomous Data Warehouse (ADW) for analytical workloads.
Autonomo
us Database provides high performance, scalability, and reliability, making
it a
n ideal choice for modern cloud-based applications.


SQL> select ai explainsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS customer_count
FROM SH.CUSTOMERS AS c
WHERE c.CUST_STATE_PROVINCE = 'San Francisco' AND c.CUST_MARITAL_STATUS =
'Married';

Explanation:
- We use the 'SH' table alias for the 'CUSTOMERS' table for better
readability.
- The query uses the 'COUNT(*)' function to count the number of rows that
match the given conditions.
- The 'WHERE' clause is used to filter the results:
  - 'c.CUST_STATE_PROVINCE = 'San Francisco'' filters customers who have 'San
Francisco' as their state or province.
  - 'c.CUST_MARITAL_STATUS = 'Married'' filters customers who have 'Married'
as their marital status.
```

```
The result of this query will give you the count of customers in San
Francisco who are married, using the column alias 'customer_count' for the
result.

Remember to adjust the table and column names based on your actual schema if
they differ from the example.

Feel free to ask if you have more questions related to SQL or database in
general.


SQL> EXEC DBMS_CLOUD_AI.DROP_PROFILE('OPENAI');

PL/SQL procedure successfully completed.
```

Example: Select AI with Cohere

The following example shows how you can use Cohere to generate SQL statements from natural language prompts.

> **Note:**
>
> Only an ADMIN user can run `EXECUTE` privileges and network ACL procedure.

```
--Grants EXECUTE privilege to ADB_USER
--
SQL>grant execute on DBMS_CLOUD_AI to ADB_USER;
--
-- Create Credential for AI provider
--
SQL> EXEC DBMS_CLOUD.CREATE_CREDENTIAL('COHERE_CRED', 'COHERE', '<your api
token>');

PL/SQL procedure successfully completed.

--
-- Grant Network ACL for Cohere endpoint
--
SQL> BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'api.cohere.ai',
         ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                             principal_name => 'ADB_USER',
                             principal_type => xs_acl.ptype_db)
    );
     END;
     /
    /

PL/SQL procedure successfully completed.

--
```

```
-- Create AI profile
--SQL> BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
      'COHERE',
      '{"provider": "cohere",
        "credential_name": "COHERE_CRED",
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "sales"},
                        {"owner": "SH", "name": "products"},
                        {"owner": "SH", "name": "countries"}]
       }');
       END;
       /

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
SQL> EXEC DBMS_CLOUD_AI.SET_PROFILE('COHERE');

PL/SQL procedure successfully completed.

--
-- Get Profile in current session
--
SQL> SELECT DBMS_CLOUD_AI.get_profile() from dual;

DBMS_CLOUD_AI.GET_PROFILE()
--------------------------------------------------------------------------------
--
"COHERE"

--
-- Use AI
--
SQL> select ai how many customers exist;

CUSTOMER_COUNT
--------------
         55500

SQL> EXEC DBMS_CLOUD_AI.DROP_PROFILE('COHERE');

PL/SQL procedure successfully completed.
```

Example: Select AI with Azure OpenAI Service API Key

The following example shows how you can enable access to Azure OpenAI Service using your API key, create an AI profile, and generate SQL from natural language prompts.

```
-- Create Credential for AI integration
--
SQL> EXEC DBMS_CLOUD.CREATE_CREDENTIAL('AZURE_CRED', 'AZUREAI', '<your api
token>');
```

```
PL/SQL procedure successfully completed.

--
-- Grant Network ACL for OpenAI endpoint
--SQL> BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => '<azure_resource_name>.openai.azure.com',
         ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                             principal_name => 'ADMIN',
                             principal_type => xs_acl.ptype_db)
       );
       END;
       /

PL/SQL procedure successfully completed.

--
-- Create AI profile
--
SQL>
BEGIN

DBMS_CLOUD_AI.CREATE_PROFILE(
      'AZUREAI',
      '{"provider": "azure",
        "azure_resource_name": "<azure_resource_name>",
        "azure_deployment_name":
"<azure_deployment_name>"

        "credential_name": "AZURE_CRED",
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "countries"},
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}],
        "conversation": "true"
       }');

END;
     /

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
SQL> EXEC DBMS_CLOUD_AI.SET_PROFILE('AZUREAI');

PL/SQL procedure successfully completed.

--
-- Get Profile in current session
--
SQL> SELECT DBMS_CLOUD_AI.get_profile() from dual;
```

```
DBMS_CLOUD_AI.GET_PROFILE()
--------------------------------------------------------------------------------
--
"AZUREAI"


--
-- Use AI
--
SQL> select ai how many customers exist;

CUSTOMER_COUNT
--------------
         55500

SQL> select ai how many customers in San Francisco are married;

MARRIED_CUSTOMERS
-----------------
               18


SQL> select ai showsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS married_customers_count
FROM SH.CUSTOMERS c
WHERE c.CUST_CITY = 'San Francisco'
  AND c.CUST_MARITAL_STATUS = 'Married'


SQL> select ai narrate what are the top 3 customers in San Francisco;

RESPONSE
--------------------------------------------------------------------------------
--
The top 3 customers in San Francisco are:

1. Hector Colven - Total amount sold: $52,025.99
2. Milburn Klemm - Total amount sold: $50,842.28
3. Gavin Xie - Total amount sold: $48,677.18


SQL> select ai chat what is Autonomous Database;

RESPONSE
--------------------------------------------------------------------------------
--
Autonomous Database is a cloud-based database service provided by Oracle. It
is
designed to automate many of the routine tasks involved in managing a
database,
such as patching, tuning, and backups. Autonomous Database uses machine
learning
```

```
 and automation to optimize performance, security, and availability, allowing
us
ers to focus on their applications and data rather than database
administration
tasks. It offers both Autonomous Transaction Processing (ATP) for
transactional
workloads and Autonomous Data Warehouse (ADW) for analytical workloads.
Autonomo
us Database provides high performance, scalability, and reliability, making
it a
n ideal choice for modern cloud-based applications.


SQL> select ai explainsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS customer_count
FROM SH.CUSTOMERS AS c
WHERE c.CUST_STATE_PROVINCE = 'San Francisco' AND c.CUST_MARITAL_STATUS =
'Married';

Explanation:
- We use the 'SH' table alias for the 'CUSTOMERS' table for better
readability.
- The query uses the 'COUNT(*)' function to count the number of rows that
match the given conditions.
- The 'WHERE' clause is used to filter the results:
  - 'c.CUST_STATE_PROVINCE = 'San Francisco'' filters customers who have 'San
Francisco' as their state or province.
  - 'c.CUST_MARITAL_STATUS = 'Married'' filters customers who have 'Married'
as their marital status.
The result of this query will give you the count of customers in San
Francisco who are married, using the column alias 'customer_count' for the
result.

Remember to adjust the table and column names based on your actual schema if
they differ from the example.

Feel free to ask if you have more questions related to SQL or database in
general.


SQL> EXEC DBMS_CLOUD_AI.DROP_PROFILE('AZUREAI');

PL/SQL procedure successfully completed.
```

Example: Select AI with Azure OpenAI Service Principle

Connect as an ADMIN to provide access to Azure service principle authentication and then grant the network ACL permissions to the user (`ADB_USER`) who wants to use Select AI. To provide access to Azure resources, see Use Azure Service Principal to Access Azure Resources.

> **Note:**
>
> Only an ADMIN user can run `EXECUTE` privileges and network ACL procedure.

```
-- Connect as ADMIN user and enable Azure service principal authentication.
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(provider  => 'AZURE',
                                         params    =>
JSON_OBJECT('azure_tenantid' value 'azure_tenantid'));
END;
/

-- Copy the consent url from cloud_integrations view and consents the ADB-S
application.
SQL> select param_value from CLOUD_INTEGRATIONS where param_name =
'azure_consent_url';
PARAM_VALUE
--------------------------------------------------------------------------------
--
https://login.microsoftonline.com/<tenant_id>/oauth2/v2.0/authorize?
client_id=<client_id>&response_type=code&scope=User.read

-- On the Azure OpenAI IAM console, search for the Azure application name and
assign the permission to the application.
-- You can get the application name in the cloud_integrations view.
SQL> select param_value from CLOUD_INTEGRATIONS where param_name =
'azure_app_name';
PARAM_VALUE
--------------------------------------------------------------------------------
--
ADBS_APP_DATABASE_OCID

--
-- Grant Network ACL for Azure OpenAI endpoint
--SQL> BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'azure_resource_name.openai.azure.com',
         ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                             principal_name => 'ADB_USER',
                             principal_type => xs_acl.ptype_db)
       );
       END;
       /

PL/SQL procedure successfully completed.

--
-- Create AI profile
--SQL>
BEGIN

DBMS_CLOUD_AI.CREATE_PROFILE(
      'AZUREAI',
      '{"provider": "azure",
```

```
        "credential_name": "AZURE$PA",
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "countries"},
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}],
        "azure_resource_name":
"<azure_resource_name>",
        "azure_deployment_name": "<azure_deployment_name>"
       }');

END;
     /

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
SQL> EXEC DBMS_CLOUD_AI.SET_PROFILE('AZUREAI');

PL/SQL procedure successfully completed.

--
-- Get Profile in current session
--
SQL> SELECT DBMS_CLOUD_AI.get_profile() from dual;

DBMS_CLOUD_AI.GET_PROFILE()
--------------------------------------------------------------------------------
--
"AZUREAI"

--
-- Use AI
--
SQL> select ai how many customers exist;

CUSTOMER_COUNT
--------------
         55500

SQL> select ai how many customers in San Francisco are married;

MARRIED_CUSTOMERS
-----------------
               18


SQL> select ai showsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS married_customers_count
```

```
FROM SH.CUSTOMERS c
WHERE c.CUST_CITY = 'San Francisco'
  AND c.CUST_MARITAL_STATUS = 'Married'


SQL> select ai narrate what are the top 3 customers in San Francisco;

RESPONSE
--------------------------------------------------------------------------------
--
The top 3 customers in San Francisco are:

1. Hector Colven - Total amount sold: $52,025.99
2. Milburn Klemm - Total amount sold: $50,842.28
3. Gavin Xie - Total amount sold: $48,677.18


SQL> select ai chat what is Autonomous Database;

RESPONSE
--------------------------------------------------------------------------------
--
Autonomous Database is a cloud-based database service provided by Oracle. It
is
designed to automate many of the routine tasks involved in managing a
database,
such as patching, tuning, and backups. Autonomous Database uses machine
learning
 and automation to optimize performance, security, and availability, allowing
us
ers to focus on their applications and data rather than database
administration
tasks. It offers both Autonomous Transaction Processing (ATP) for
transactional
workloads and Autonomous Data Warehouse (ADW) for analytical workloads.
Autonomo
us Database provides high performance, scalability, and reliability, making
it a
n ideal choice for modern cloud-based applications.


SQL> select ai explainsql how many customers in San Francisco are married;

RESPONSE
--------------------------------------------------------------------------------
--
SELECT COUNT(*) AS customer_count
FROM SH.CUSTOMERS AS c
WHERE c.CUST_STATE_PROVINCE = 'San Francisco' AND c.CUST_MARITAL_STATUS =
'Married';

Explanation:
- We use the 'SH' table alias for the 'CUSTOMERS' table for better
readability.
- The query uses the 'COUNT(*)' function to count the number of rows that
match the given conditions.
```

```
- The 'WHERE' clause is used to filter the results:
  - 'c.CUST_STATE_PROVINCE = 'San Francisco'' filters customers who have 'San
Francisco' as their state or province.
  - 'c.CUST_MARITAL_STATUS = 'Married'' filters customers who have 'Married'
as their marital status.
The result of this query will give you the count of customers in San
Francisco who are married, using the column alias 'customer_count' for the
result.

Remember to adjust the table and column names based on your actual schema if
they differ from the example.

Feel free to ask if you have more questions related to SQL or database in
general.


SQL> EXEC DBMS_CLOUD_AI.DROP_PROFILE('AZUREAI');

PL/SQL procedure successfully completed.
```

Select AI with OCI Generative AI API Key

The following example shows how you can access OCI Generative AI using your OCI API key, create an AI profile, and generate SQL from natural language prompts.

> **Note:**
>
> OCI Generative AI uses `cohere.command` as the default model if you do not specify the `model_name`. To learn more about the parameters, see Profile Attributes.

```
-- Create Credential with OCI API key
--
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'GENAI_CRED',
    user_ocid       => 'ocid1.user.oc1..aaaa...',
    tenancy_ocid    => 'ocid1.tenancy.oc1..aaaa...',
    private_key     => '<your_api_key>',
    fingerprint     => '<your_fingerprint>'
  );
END;
/

--
-- Create AI profile
--
SQL>
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
      'GENAI',
      '{"provider":
"oci",
        "credential_name": "GENAI_CRED"
```

```
      }');

END;
     /

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
--
SQL> EXEC DBMS_CLOUD_AI.SET_PROFILE('GENAI');

PL/SQL procedure successfully completed.

--
-- Get Profile in current session
--
SQL> SELECT DBMS_CLOUD_AI.get_profile() from dual;

DBMS_CLOUD_AI.GET_PROFILE()
--------------------------------------------------------------------------------
--
"GENAI"

--
-- Use AI
--

SQL> select ai chat what is Autonomous Database;

RESPONSE
--------------------------------------------------------------------------------
--
Autonomous Database is a cloud-based database service provided by Oracle. It
is
designed to automate many of the routine tasks involved in managing a
database,
such as patching, tuning, and backups. Autonomous Database uses machine
learning
 and automation to optimize performance, security, and availability, allowing
us
ers to focus on their applications and data rather than database
administration
tasks. It offers both Autonomous Transaction Processing (ATP) for
transactional
workloads and Autonomous Data Warehouse (ADW) for analytical workloads.
Autonomo
us Database provides high performance, scalability, and reliability, making
it a
n ideal choice for modern cloud-based applications.


SQL> EXEC DBMS_CLOUD_AI.DROP_PROFILE('GENAI');

PL/SQL procedure successfully completed.
```

Select AI with OCI Generative AI Resource Principal

To use resource principal with OCI Generative AI, Oracle Cloud Infrastructure tenancy administrator must grant access for Generative AI resources to a dynamic group. See Perform Prerequisites to Use Resource Principal with Autonomous Database to provide access to a dynamic group.

Set the required policies to obtain access to all Generative AI resources. See https://docs.oracle.com/en-us/iaas/Content/generative-ai/iam-policies.htm to know more about Generative AI policies.

- To get access to all Generative AI resources in the entire tenancy, use the following policy:

  ```
  allow group <your-group-name> to manage generative-ai-family in tenancy
  ```

- To get access to all Generative AI resources in your compartment, use the following policy:

  ```
  allow group <your-group-name> to manage generative-ai-family in
  compartment <your-compartment-name>
  ```

Connect as an administrator and enable OCI resource principal. See ENABLE_PRINCIPAL_AUTH Procedure to configure the parameters.

> **Note:**
>
> OCI Generative AI uses `cohere.command` as the default model if you do not specify the `model_name`. To learn more about the parameters, see Profile Attributes.

```
-- Connect as ADMIN user and enable OCI resource principal.
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(provider  => 'OCI');
END;
/

--
-- Create AI profile
--
SQL>BEGIN

  DBMS_CLOUD_AI.CREATE_PROFILE(
      'GENAI',
      '{"provider":
"oci",
        "credential_name": "OCI$RESOURCE_PRINCIPAL"
       }');

END;
    /

PL/SQL procedure successfully completed.

--
-- Enable AI profile in current session
```

```
--
SQL>EXEC DBMS_CLOUD_AI.SET_PROFILE('GENAI');

PL/SQL procedure successfully completed.

--
-- Get Profile in current session
--
SQL> SELECT DBMS_CLOUD_AI.get_profile() from dual;

DBMS_CLOUD_AI.GET_PROFILE()
--------------------------------------------------------------------------------
--
"GENAI"

-- Use AI

SQL> select ai chat what is Autonomous Database;

RESPONSE
--------------------------------------------------------------------------------
--
Autonomous Database is a cloud-based database service provided by Oracle. It
is
designed to automate many of the routine tasks involved in managing a
database,
such as patching, tuning, and backups. Autonomous Database uses machine
learning
 and automation to optimize performance, security, and availability, allowing
us
ers to focus on their applications and data rather than database
administration
tasks. It offers both Autonomous Transaction Processing (ATP) for
transactional
workloads and Autonomous Data Warehouse (ADW) for analytical workloads.
Autonomo
us Database provides high performance, scalability, and reliability, making
it a
n ideal choice for modern cloud-based applications.



SQL> EXEC DBMS_CLOUD_AI.DROP_PROFILE('GENAI');

PL/SQL procedure successfully completed.
```

Select AI with OCI Generative AI Using LLMA Model

This example showcases the `chat` feature of the LLMA model from OCI Generative AI. It highlights the model's capabilities through two prompts: generating a friendly email to customers for movie recommendations and generate an introductory paragraph on rock climbing.

```
SQL>
BEGIN
```

```
DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name =>
'GENAI_CRED',
        user_ocid       => 'ocid1.user.oc1..aaa',
        tenancy_ocid    => 'ocid1.tenancy.oc1..aaa',
        private_key     => '<your_api_key>',
        fingerprint     => '<your_fingerprint>'
      );


END;

    /

    PL/SQL procedure successfully completed.


BEGIN


DBMS_CLOUD_AI.CREATE_PROFILE(
          'GENAI',
          '{"provider": "oci",
            "model": "meta.llama-2-70b-chat",
            "oci_runtimetype":"LLAMA"}');

END;

    /

    PL/SQL procedure successfully completed.

SQL>
BEGIN


DBMS_CLOUD_AI.SET_ATTRIBUTE(
          'GENAI', 'credential_name',
'GENAI_CRED');

END;

    /


    PL/SQL procedure successfully completed.

SQL >
BEGIN


DBMS_CLOUD_AI.SET_ATTRIBUTE(
          'GENAI', 'oci_compartment_id', 'ocid1.compartment.oc1...');

END;
```

```
      /

    PL/SQL procedure successfully completed.

SQL> EXEC DBMS_CLOUD_AI.SET_PROFILE('GENAI');

    PL/SQL procedure successfully completed.

SQL> set linesize
150
SQL> SELECT AI chat Generate a friendly email to customer Gilbert Lee with
two action-thriller movie recommendations available through our MovieStream
service;
SQL>
    RESPONSE

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
    .
    Subject: Action-packed movie recommendations for you!
    Dear Gilbert,
    I hope this email finds you well! I wanted to reach out to you today to
recommend two action-thriller movies that are currently available on our Movie
    Stream service. I think you'll really enjoy them!
    The first movie I recommend is "John Wick" starring Keanu Reeves. This
movie follows the story of a retired hitman who seeks vengeance against a
power
    ful crime lord and his army of assassins. The action scenes are intense
and non-stop, and Keanu Reeves delivers an outstanding performance.
    RESPONSE

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
    The second movie I recommend is "Mission: Impossible - Fallout" starring
Tom Cruise. This movie follows Ethan Hunt and his team as they try to prevent
     a global catastrophe. The action scenes are heart-stopping and the
stunts are truly impressive. Tom Cruise once again proves why he's one of the
grea
    test action stars of all time.
    Both of these movies are sure to keep you on the edge of your seat and
provide plenty of thrills and excitement. They're available to stream now on
Mo
    vieStream, so be sure to check them out!
    If you have any questions or need assistance with MovieStream, please
don't hesitate to reach out to me. I'm always here to help.
    Thank you for being a valued customer, and I hope you enjoy the movies!
    RESPONSE

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
    Best regards,
    [Your Name]
    MovieStream Customer Service

SQL> SELECT AI chat Write an enthusiastic introductory paragraph on how to
```

```
get started with rock climbing with Athletes as the target audience;
    RESPONSE

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
    Rock climbing is an exhilarating and challenging sport that's perfect for
athletes looking to push their limits and test their strength, endurance, an
    d mental toughness. Whether you're a seasoned athlete or just starting
out, rock climbing offers a unique and rewarding experience that will have you
    hooked from the very first climb. With its combination of physical and
mental challenges, rock climbing is a great way to build strength, improve
flex
    ibility, and develop problem-solving skills. Plus, with the supportive
community of climbers and the breathtaking views from the top of the climb,
you
    'll be hooked from the very first climb. So, if you're ready to take on a
new challenge and experience the thrill of adventure, then it's time to get
    started with rock climbing!
```

Improve SQL Query Generation by Adding Comments to Database Tables and Columns

This example demonstrates how comments in database tables and columns can improve the generation of SQL queries from natural language prompts. In this example, Azure OpenAI Service acts as the AI provider. The `"comments":"true"` parameter in `DBMS_CLOUD_AI.CREATE_PROFILE` function determines whether comments are passed to the model for SQL generation.

```
-- Adding comments to table 1, table 2, and table 3. Table 1 has 3 columns,
table 2 has 7 columns, table 3 has 2 columns.

-- TABLE1
COMMENT ON TABLE table1 IS 'Contains movies, movie titles and the year it was
released';
COMMENT ON COLUMN table1.c1 IS 'movie ids. Use this column to join to other
tables';
COMMENT ON COLUMN table1.c2 IS 'movie titles';
COMMENT ON COLUMN table1.c3 IS 'year the movie was released';
-- TABLE2
COMMENT ON TABLE table2 IS 'transactions for movie views - also known as
streams';
COMMENT ON COLUMN table2.c1 IS 'day the movie was streamed';
COMMENT ON COLUMN table2.c2 IS 'genre ids. Use this column to join to other
tables';
COMMENT ON COLUMN table2.c3 IS 'movie ids. Use this column to join to other
tables';
COMMENT ON COLUMN table2.c4 IS 'customer ids. Use this column to join to
other tables';
COMMENT ON COLUMN table2.c5 IS 'device used to stream, watch or view the
movie';
COMMENT ON COLUMN table2.c6 IS 'sales from the movie';
COMMENT ON COLUMN table2.c7 IS 'number of views, watched, streamed';

-- TABLE3
COMMENT ON TABLE table3 IS 'Contains the genres';
COMMENT ON COLUMN table3.c1 IS 'genre id. use this column to join to other
tables';
```

```
COMMENT ON COLUMN table3.c2 IS 'name of the genre';


BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'myprofile',
    attributes =>
        '{"provider": "azure",
          "azure_resource_name": "my_resource",
          "azure_deployment_name": "my_deployment",
          "credential_name": "my_credential",
          "comments":"true",
          "object_list": [
            {"owner": "moviestream", "name": "table1"},
            {"owner": "moviestream", "name": "table2"},
            {"owner": " moviestream", "name": "table3"}
          ]
          }'
    );

    DBMS_CLOUD_AI.SET_PROFILE(
        profile_name => 'myprofile'
    );

END;
/

--Prompts
select ai what are our total views;
RESPONSE
--------------------------------------------------
TOTAL_VIEWS
-----------
   97890562

select ai showsql what are our total views;

RESPONSE
--------------------------------------------------------------------------------
SELECT SUM(QUANTITY_SOLD) AS total_views
FROM "moviestream"."table"

select ai what are our total views broken out by device;

DEVICE                    TOTAL_VIEWS
------------------------- -----------
mac                          14719238
iphone                       20793516
ipad                         15890590
pc                           14715169
galaxy                       10587343
pixel                        10593551
lenovo                        5294239
fire                          5296916

8 rows selected.
```

```
select ai showsql what are our total views broken out by device;

RESPONSE

--------------------------------------------------------------------------------
---------
SELECT DEVICE, COUNT(*) AS TOTAL_VIEWS
FROM "moviestream"."table"
GROUP BY DEVICE
```

Terminology

Definition of some of the terms used in Select AI feature are described.

The following are the terms related to Select AI feature:

Term	Definition
Database Credential	Database Credentials are authentication credentials used to access and interact with databases. They typically consist of a user name and a password, sometimes supplemented by additional authentication factors like security tokens. These credentials are used to establish a secure connection between an application or user and a database, ensuring that only authorized individuals or systems can access and manipulate the data stored within the database.
Hallucination in LLM	Hallucination in the context of Large Language Models refers to a phenomenon where the model generates text that is incorrect, nonsensical, or unrelated to the input prompt. Despite being a result of the model's attempt to generate coherent text, these instances can contain information that is fabricated, misleading, or purely imaginative. Hallucination can occur due to biases in training data, lack of proper context understanding, or limitations in the model's training process.
IAM	Oracle Cloud Infrastructure Identity and Access Management (IAM) lets you control who has access to your cloud resources. You can control what type of access a group of users have and to which specific resources. To learn more, see Overview of Identity and Access Management.
Large Language Model (LLM)	Large Language Models refer to advanced artificial intelligence models that are trained on massive amounts of text data to understand and generate human-like language, software code, and database queries. These models are capable of performing a wide range of natural language processing tasks, including text generation, translation, summarization, question answering, sentiment analysis, and more. LLMs are typically neural network-based architectures that learn patterns, context, and semantics from the input data, enabling them to generate coherent and contextually relevant text.
Natural Language Prompts	Natural Language Prompts are human-readable instructions or requests provided to guide generative AI models, such as Large Language Models. Instead of using specific programming languages or commands, users can interact with these models by entering prompts in a more conversational or natural language form. The models then generate output based on the provided prompt.

Term	Definition
Network Access Control List (ACL)	A Network Access Control List is a set of rules or permissions that define what network traffic is allowed to pass through a network device, such as a router, firewall, or gateway. ACLs are used to control and filter incoming and outgoing traffic based on various criteria such as IP addresses, port numbers, and protocols. They play a crucial role in network security by enabling administrators to manage and restrict network traffic to prevent unauthorized access, potential attacks, and data breaches.

JSON Document Stores

Autonomous Database has full support for data represented as JSON documents. In Autonomous Databases, JSON documents can coexist with relational data.

For more details on JSON Documents, see JSON Developer's Guide.

Cache Messages with Singleton Pipes and Using Pipes for Persistent Messaging

On Autonomous Database the `DBMS_PIPE` package includes support for Singleton Pipes and Persistent Messaging Pipes.

- **Singleton Pipes** allow you to cache and retrieve a custom message and share the message across multiple database sessions with concurrent reads.
- **Persistent Messaging Pipes** allow you to store messages in Cloud Object Storage. Using persistent messages you can send and receive messages within a single database or between multiple databases in the same or in different regions, without message size limitations.
- Cache Messages with Singleton Pipes
  Singleton Pipe is an addition to the `DBMS_PIPE` package that allows you to cache and retrieve a custom message and share the message across multiple database sessions with concurrent reads.
- Use Persistent Messaging with Messages Stored in Cloud Object Store
  The `DBMS_PIPE` package has extended functionality on Autonomous Database to support persistent messaging, where messages are stored in Cloud Object Store.

Cache Messages with Singleton Pipes

Singleton Pipe is an addition to the `DBMS_PIPE` package that allows you to cache and retrieve a custom message and share the message across multiple database sessions with concurrent reads.

- About Caching Messages with Singleton Pipes
  The `DBMS_PIPE` package has extended functionality on Autonomous Database to support Singleton Pipes.
- Automatic Refresh of Cached Message with a Cache Function
  The `DBMS_PIPE` package allows you to automatically populate a Singleton Pipe message using a user-defined cache function.

- Create an Explicit Singleton Pipe
  Describes the steps to create a Singleton Pipe with a specified pipe name (an Explicit Singleton Pipe).
- Create an Explicit Singleton Pipe with a Cache Function
  Describes the steps to create a Singleton Pipe with a specified pipe name, an Explicit Singleton Pipe, and provide a cache function. A cache function allows you to automatically populate the message in a singleton pipe.

About Caching Messages with Singleton Pipes

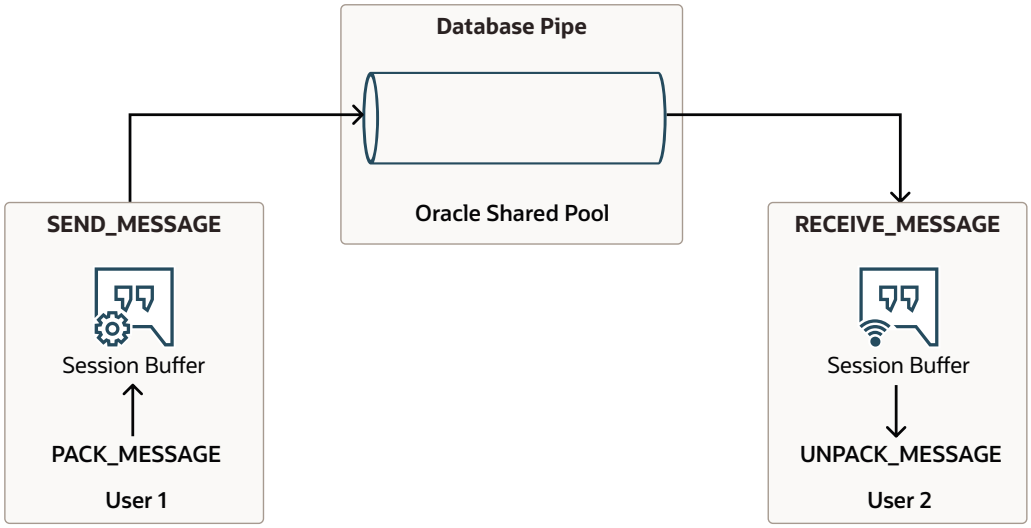
The `DBMS_PIPE` package has extended functionality on Autonomous Database to support Singleton Pipes.

A Singleton Pipe in `DBMS_PIPE`:

- Provides in-memory caching of custom data using Singleton Pipe messages.
- Supports the ability to cache and retrieve a custom message of up to 32,767 bytes.
- Supports sharing a cached message across multiple database sessions with concurrent reads. This provides high throughput and supports concurrent reads of messages across database sessions.
- Supports Read-Only and Read-Write databases.
- Supports several cache invalidation methods:
  - Explicit cache invalidation controlled by user.
  - Cache invalidation after a user specified time interval (in seconds). This invalidation method is controlled by the message sender, using the `shelflife` parameter, instead of by message readers. This avoids the common pitfalls due to incorrect use of cache by readers.

About Standard Pipes and Singleton Pipes

The DBMS_PIPE Package allows two or more database sessions to communicate using in-memory messages. Pipe functionality has several applications such as external service interface, debugging, independent transactions, and alerts.

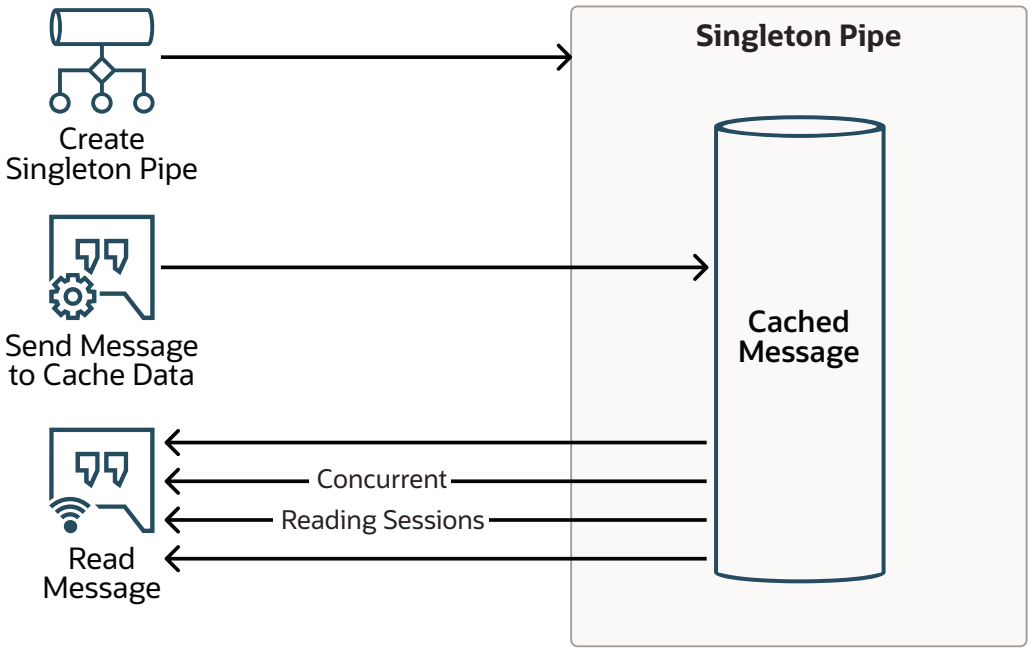


A Singleton Pipe can be any one of the supported `DBMS_PIPE` types:

- **Implicit Pipe:** Automatically created when a message is sent with an unknown pipe name using the `DBMS_PIPE.SEND_MESSAGE` function.
- **Explicit Pipe:** Created using the `DBMS_PIPE.CREATE_PIPE` function with a user specified pipe name.
- **Public Pipe:** Accessible by any user with `EXECUTE` permission on `DBMS_PIPE` package.
- **Private Pipe:** Accessible by sessions with the same user as the pipe creator.

Singleton Pipes provide the ability to cache a single message in the memory of the Autonomous Database instance.

The following shows the general workflow for using singleton pipes.



Singleton Pipe Overview and Features

- **Singleton Messages**
  - A Singleton Pipe can cache one message in the pipe, hence the name "singleton".
  - The message in a Singleton Pipe can be comprised of multiple fields, up to a total message size of 32,767 bytes.
  - `DBMS_PIPE` supports the ability to pack multiple attributes in a message using `DBMS_PIPE.PACK_MESSAGE` procedure.
  - For a Public Singleton Pipe, the message can be received by any database session with execute privilege on `DBMS_PIPE` package.
  - For Private Singleton Pipe, the message can be received by sessions with the same user as the creator of the Singleton Pipe.
- **High Message Throughput for Reads**
  - Singleton Pipes cache the message in the pipe until it is invalidated or purged. Database sessions can concurrently read a message from the Singleton Pipe.
  - Receiving a message from a Singleton Pipe is a non-blocking operation.
- **Message Caching**

  - A message is cached in a Singleton Pipe using `DBMS_PIPE.SEND_MESSAGE`.
  - If there is an existing cached message in the Singleton Pipe, then `DBMS_PIPE.SEND_MESSAGE` overwrites the previous message to maintain only one message in the Singleton Pipe.
- **Message Invalidation**
  - **Explicit Invalidation**: purges the pipe with the procedure `DBMS_PIPE.PURGE` or by overwriting the message using `DBMS_PIPE.SEND_MESSAGE`.
  - **Automatic Invalidation**: a message can be invalidated automatically after the specified `shelflife` time has elapsed.
- **No Eviction from Database Memory**
  - Singleton Pipes do not get evicted from Oracle Database memory.
  - An Explicit Singleton Pipe continues to reside in database memory until it is removed using `DBMS_PIPE.REMOVE_PIPE` or until the database restarts.
  - An Implicit Singleton Pipe stays in database memory until there is one cached message in the pipe.

Singleton Pipe Operations

Operation	DBMS_PIPE Function or Procedure
Create an Explicit Singleton Pipe	CREATE_PIPE Function
Cache a message in Singleton Pipe	PACK_MESSAGE Procedures, SEND_MESSAGE Function
Read a cached message from Singleton Pipe	RECEIVE_MESSAGE Function, UNPACK_MESSAGE Procedures
Delete a message in Singleton Pipe	PURGE Procedure
Remove an Explicit Singleton Pipe	REMOVE_PIPE Function

Automatic Refresh of Cached Message with a Cache Function

The `DBMS_PIPE` package allows you to automatically populate a Singleton Pipe message using a user-defined cache function.

By default, after a message is invalidated with either Singleton Pipe explicit or implicit invalidation, a subsequent `DBMS_PIPE.RECEIVE_MESSAGE` results in no message being received. To add a new message to the pipe, the message must be explicitly cached by calling `DBMS_PIPE.SEND_MESSAGE`. To avoid this case, where no message is available when you read from a Singleton Pipe, you can define a cache function. With a cache function defined, the cache function is automatically invoked when you receive a message in following scenarios:

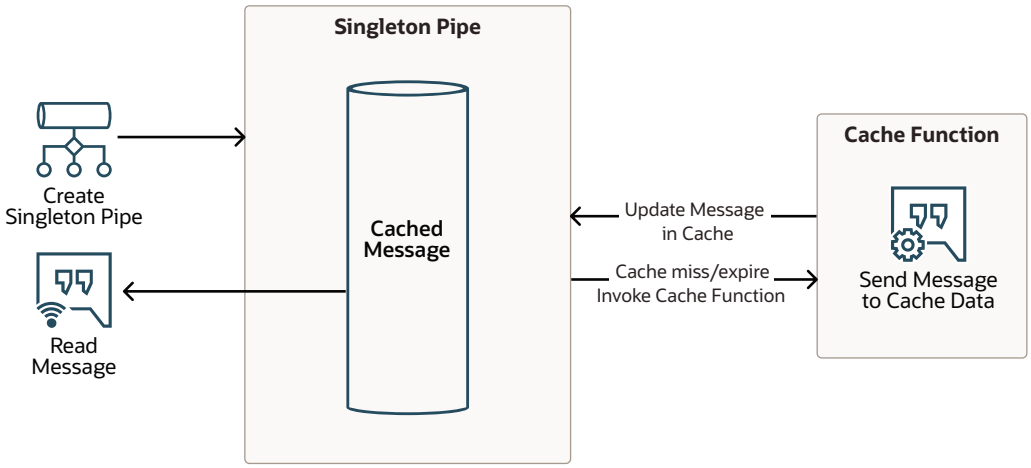
- When the Singleton Pipe is empty.
- When the message in a Singleton Pipe is invalid due to the `shelflife` time elapsed.

To use a cache function define the cache function and include the `cache_func` parameter with `DBMS_PIPE.RECEIVE_MESSAGE`. A user-defined cache function provides the following:

- The cache function can be specified when reading a message from a Singleton Pipe using `DBMS_PIPE.RECEIVE_MESSAGE`.
- When there is no message in the Singleton Pipe, `DBMS_PIPE.RECEIVE_MESSAGE` calls the cache function.

- When the message `shelflife` time has elapsed, the database automatically populates a new message in the Singleton Pipe.

Using a cache function simplifies working with Singleton Pipes. You do not need to handle failure cases for receiving a message from an empty pipe. In addition, a cache function ensures there is no cache-miss when you read messages from a Singleton Pipe, providing maximum use of the cached message.



When you define a cache function, the function name must be fully qualified with the owner schema:

- *`OWNER.FUNCTION_NAME`*
- *`OWNER.PACKAGE.FUNCTION_NAME`*

Define a cache function with the following signature:

```
CREATE OR REPLACE FUNCTION cache_function_name(
       pipename  IN VARCHAR2
) RETURN INTEGER;
```

The typical operations within a cache function are:

- Create a Singleton Pipe, for an Explicit Pipe, using `DBMS_PIPE.CREATE_PIPE`.
- Create a message to cache in the Singleton Pipe.
- Send the message to the pipe specified in the cache function, optionally specifying a `shelflife` for the implicit message.

To use a cache function, the current session user that invokes `DBMS_PIPE.RECEIVE_MESSAGE` must have required privileges to execute the cache function.

See RECEIVE_MESSAGE Function more information on defining a cache function.

Create an Explicit Singleton Pipe

Describes the steps to create a Singleton Pipe with a specified pipe name (an Explicit Singleton Pipe).

First, for this example create the `receive_message` helper function to repeatedly call `DBMS_PIPE.RECEIVE_MESSAGE`. This allows you to test singleton pipe functionality.

```
CREATE OR REPLACE FUNCTION msg_types AS
       TYPE t_rcv_row IS RECORD (c1 VARCHAR2(32767), c2 NUMBER);
       TYPE t_rcv_tab IS TABLE OF t_rcv_row;
END;


CREATE OR REPLACE FUNCTION receive_message(
      pipename    IN VARCHAR2,
      rcv_count   IN NUMBER DEFAULT 1,
      cache_func  IN VARCHAR2 DEFAULT NULL)
   RETURN msg_types.t_rcv_tab pipelined
    AS
       l_msg    VARCHAR2(32767);
       l_status NUMBER;
 BEGIN
      FOR i IN 1..rcv_count LOOP
           l_status := DBMS_PIPE.RECEIVE_MESSAGE(
            pipename   => pipename,
            cache_func => cache_func,
            timeout    => 1);
         IF l_status != 0 THEN
              raise_application_error(-20000,
             'Message not received for attempt: ' || to_char(i) || ' status:
' ||
            l_status);
         END IF;

         DBMS_PIPE.UNPACK_MESSAGE(l_msg);
             pipe row(msg_types.t_rcv_row(l_msg));
     END LOOP;
 RETURN;
 END;
```

1. Create an explicit singleton pipe named `PIPE_TEST` with `shelflife` parameter set to 3600 (seconds).

   ```
   DECLARE
     l_status INTEGER;
   BEGIN
     l_status := DBMS_PIPE.CREATE_PIPE(
                   pipename => 'MY_PIPE1',
                   private => TRUE,
                   singleton => TRUE,
                   shelflife => 3600);
   END;
   /
   ```

See CREATE_PIPE Function for more information.

2. Verify the singleton pipe is created.

```
SELECT name, singleton, type
     FROM v$db_pipes WHERE name= '&pipename' ORDER BY 1;

NAME                 SINGLETON  TYPE
-------------------- ---------- -------
PIPE_TEST            YES        PRIVATE
```

3. Pack and send a message on the singleton pipe.

```
EXEC DBMS_PIPE.PACK_MESSAGE('This is a real message that you can get
multiple times');

SELECT DBMS_PIPE.SEND_MESSAGE(pipename => '&pipename') status FROM DUAL;

STATUS
----------
0
```

See PACK_MESSAGE Procedures and SEND_MESSAGE Function for more information.

4. Receive a message from a singleton pipe.

```
SELECT * FROM receive_message(
    pipename => '&pipename',
    rcv_count => 2);

MESSAGE
--------------------------------------------------------------------------------
-----
This is a real message that you can get multiple times
This is a real message that you can get multiple times
```

The `receive_message` function is a helper function that calls `DBMS_PIPE.RECEIVE_MESSAGE`.

5. Purge the message and remove the pipe.

```
EXEC DBMS_PIPE.PURGE('&pipename');
SELECT DBMS_PIPE.REMOVE_PIPE('&pipename') status FROM DUAL;
```

Create an Explicit Singleton Pipe with a Cache Function

Describes the steps to create a Singleton Pipe with a specified pipe name, an Explicit Singleton Pipe, and provide a cache function. A cache function allows you to automatically populate the message in a singleton pipe.

1. Create a cache function, `test_cache_message` for a singleton pipe.

```
CREATE OR REPLACE FUNCTION test_cache_message(
     pipename IN VARCHAR2) return NUMBER
AS
```

```
   l_status NUMBER;
   l_data VARCHAR2(4000);
BEGIN
   l_status := DBMS_PIPE.CREATE_PIPE(
          pipename => pipename,
          private => TRUE,
          singleton => true,
          shelflife => 600);
   IF l_status != 0 THEN RETURN l_status;
   END IF;

   DBMS_PIPE.PACK_MESSAGE('This is a placeholder cache message for an
empty pipe');
   l_status := DBMS_PIPE.SEND_MESSAGE(pipename => pipename);
   RETURN l_status;
 END;
/
```

> **Note:**
>
> The current session user invoking `DBMS_PIPE.RECEIVE_MESSAGE` must have required privilege to execute the cache function.

2. Receive with a cache function and confirm the message populates in pipe. The pipe must exist as a private pipe created in the cache function.

```
SELECT * FROM receive_message(
     pipename => '&pipename',
     rcv_count => 1,
     cache_func => 'TEST_CACHE_MESSAGE');

MESSAGE
---------------
This is a placeholder cache message for an empty pipe
```

The `receive_message` function is a helper function that calls `DBMS_PIPE.RECEIVE_MESSAGE`. See Create an Explicit Singleton Pipe for the `receive_message` definition.

See CREATE_PIPE Function for more information.

3. Receive without the cache function to confirm the message persists in the pipe.

```
SELECT * FROM receive_message(
     pipename => '&pipename',
     rcv_count => 2);

MESSAGE
---------------
This is a placeholder cache message for an empty pipe
This is a placeholder cache message for an empty pipe
```

The `receive_message` function is a helper function that calls `DBMS_PIPE.RECEIVE_MESSAGE`. See Create an Explicit Singleton Pipe for the `receive_message` definition.

See CREATE_PIPE Function for more information.

Use Persistent Messaging with Messages Stored in Cloud Object Store

The DBMS_PIPE package has extended functionality on Autonomous Database to support persistent messaging, where messages are stored in Cloud Object Store.

- About Persistent Messaging with DBMS_PIPE
- Create an Explicit Persistent Pipe and Send a Message
  Describes the steps to create a persistent pipe with a specified pipe name (Explicit Pipe).
- Retrieve a Persistent Message on Same Database
  Describes the steps to retrieve a persistent message from an explicit pipe on the same Autonomous Database instance (the instance where the message was sent).
- Retrieve a Persistent Message by Creating a Pipe on a Different Database
  Describes the steps to retrieve a persistent message stored in Cloud Object Store with an explicit pipe on an Autonomous Database instance that is different than the instance that sent the message.
- Remove a Persistent Pipe
  Describes the steps to remove a persistent pipe.

About Persistent Messaging with DBMS_PIPE

Persistent messaging with DBMS_PIPE allows one or more database sessions to communicate in the same region or across regions with messages that are stored in Cloud Object Store. Persistent messages in DBMS_PIPE:

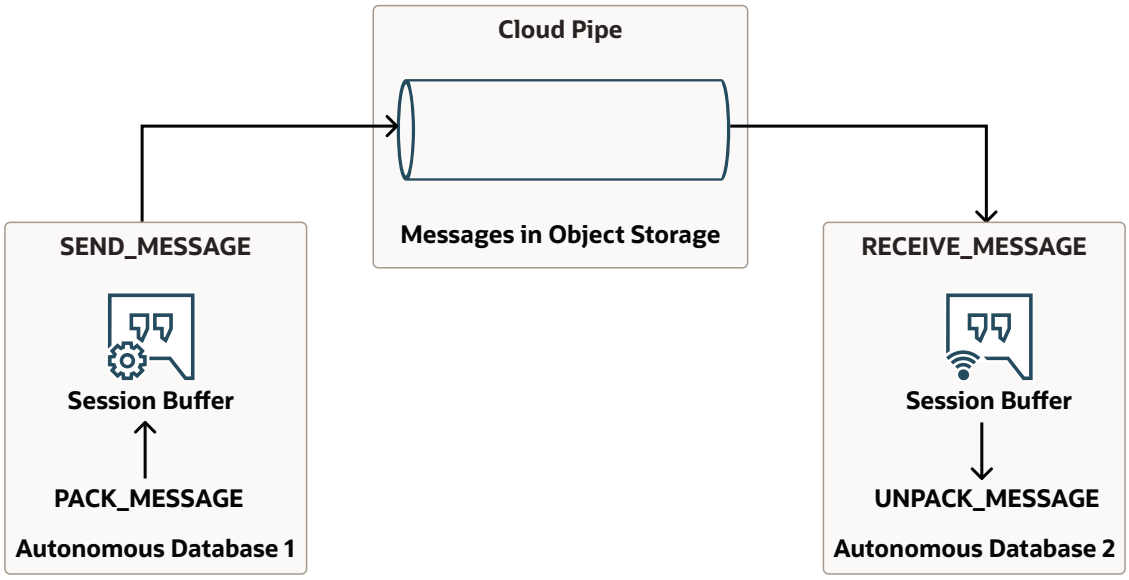
- Allow you to send and retrieve very large messages.
- Support a sending a large number of pipe messages.
- Support sending and receiving messages within a single database, across multiple databases and across databases in different regions.
- Support multiple pipes using the same Cloud Object Store location URI.

Persistent messaging pipes can be created in any of the supported DBMS_PIPE types:

- **Implicit Pipe:** Automatically created when a message is sent with an unknown pipe name using the DBMS_PIPE.SEND_MESSAGE function.
- **Explicit Pipe:** Created using the DBMS_PIPE.CREATE_PIPE function with a user specified pipe name.
- **Public Pipe:** Accessible by any user with EXECUTE permission on DBMS_PIPE package.
- **Private Pipe:** Accessible by sessions with the same user as the pipe creator.

> **Note:**
>
> Oracle recommends creating an explicit pipe before you send or receive messages with persistent messaging. Creating an explicit pipe with DBMS_PIPE.CREATE_PIPE ensures that the pipe is created with the access permissions you want, either public or private (by setting the private parameter).

The following shows the general workflow for DBMS_PIPE with persistent messaging:



Existing applications using `DBMS_PIPE` can continue to operate with minimal changes. You can configure existing applications that use `DBMS_PIPE` with a credential object and location URI using a logon trigger or using some other initialization routine. After setting the `DBMS_PIPE` credential and location URI, no other changes are needed to use persistent messaging. All subsequent use of the pipe stores the messages in Cloud Object Store instead of in database memory. This allows you to change the storage method for messages from in-memory to persistent Cloud Object Storage, with minimal changes.

Persistent Messaging Overview and Features

Features of `DBMS_PIPE` using persistent messaging:

- Messages can be sent and retrieved across multiple Autonomous Database instances in the same region or across regions.
- Persistent messages are guaranteed to either be written or read by exactly one process. This prevents message content inconsistency due to concurrent writes and reads. Using a persistent messaging pipe, `DBMS_PIPE` allows only one operation, sending a message or a receiving message to be active at a given time. However, if an operation is not possible due to an ongoing operation, the process retries periodically until the `timeout` value is reached.
- `DBMS_PIPE` uses `DBMS_CLOUD` to access Cloud Object Store. Messages can be stored in any of the supported Cloud Object Stores. See DBMS_CLOUD Package File URI Formats for more information.
- `DBMS_PIPE` uses `DBMS_CLOUD` to access Cloud Object Store and all supported credential types are available:
  - `DBMS_CLOUD.CREATE_CREDENTIAL`: See CREATE_CREDENTIAL Procedure for more information.
  - Policy and role based credentials: See Accessing Cloud Resources by Configuring Policies and Roles for more information.

DBMS_PIPE Privileges Authorization and Security

The `DBMS_PIPE` procedures run with invoker's rights. Private pipes are owned by the current user and a private pipe that is created by a user can only be used by the same user. This applies to both in-memory pipes and persistent messaging pipes where messages are stored to Cloud Object Store. Sending and receiving messages run in the invoker's schema.

Using private pipes, where messages are stored to Cloud Object Store, a credential object is required for authentication with the Cloud Object store identified by the `location_uri` parameter. The invoking user must have `EXECUTE` privilege on the credential object specified with the `credential_name` parameter that is used to access the Object Store.

To use a public pipe, the user, database session, must have execute privilege on `DBMS_PIPE`. For a public pipe using persistent messaging and storing messages to Cloud Object Store, the user, database session, must have execute privilege on `DBMS_CLOUD` and execute privilege on the credential object (or you can create a credential object that is allowed to access the location URI that contains the message.

Create an Explicit Persistent Pipe and Send a Message

Describes the steps to create a persistent pipe with a specified pipe name (Explicit Pipe).

1. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`. For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'my_persistent_pipe_cred',
    username => 'adb_user@example.com',
    password => 'password'
  );
END;
/
```

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. After you store the credentials you can then use the same credential name to access Cloud Object Store to send and receive messages with `DBMS_PIPE`.

   For detailed information about the parameters, see CREATE_CREDENTIAL Procedure. For Oracle Cloud Infrastructure Object Storage, it is required that the credential uses native Oracle Cloud Infrastructure authentication.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

   > **Note:**
   >
   > Some tools like SQL*Plus and SQL Developer use the ampersand character (`&`) as a special character. If you have the ampersand character in your password use the `SET DEFINE OFF` command in those tools as shown in the example to disable the special character and get the credential created properly.

2. Create an explicit pipe to send and retrieve messages. For example, create a pipe named `ORDER_PIPE`.

```
DECLARE
  r_status INTEGER;
BEGIN
    r_status := DBMS_PIPE.CREATE_PIPE(pipename => 'ORDER_PIPE');
```

```
END;
/
```

See CREATE_PIPE Function for more information.

3. Verify that the pipe is created.

```
SELECT ownerid, name, type FROM v$db_pipes
       WHERE name = 'ORDER_PIPE';


OWNERID NAME       TYPE
------- ---------- -------
     80 ORDER_PIPE PRIVATE
```

4. Use `DBMS_PIPE` procedures to set the default access credential and location URI to store persistent messages to Cloud Object Store.

```
BEGIN
    DBMS_PIPE.SET_CREDENTIAL_NAME('my_persistent_pipe_cred');
    DBMS_PIPE.SET_LOCATION_URI('https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/');
END;
/
```

These procedures set the default credential name and the default location URI for use with `DBMS_PIPE` procedures.

If you use Oracle Cloud Infrastructure Object Storage to store messages, you can use Oracle Cloud Infrastructure Native URIs or Swift URIs. However, the location URI and the credential must match in type as follows:

- If you use a native URI format to access Oracle Cloud Infrastructure Object Storage, you must use Native Oracle Cloud Infrastructure Signing Keys authentication in the credential object.
- If you use Swift URI format to access Oracle Cloud Infrastructure Object Storage, you must use an auth token authentication in the credential object.

See SET_CREDENTIAL_NAME Procedure and SET_LOCATION_URI Procedure for more information.

5. Pack and send a message on the pipe.

```
DECLARE
  l_result INTEGER;
  l_date   DATE;
BEGIN
    l_date := sysdate;
    DBMS_PIPE.PACK_MESSAGE(l_date);         -- date of order
    DBMS_PIPE.PACK_MESSAGE('C123');         -- order number
    DBMS_PIPE.PACK_MESSAGE(5);              -- number of items in order
    DBMS_PIPE.PACK_MESSAGE('Printers');     -- type of item in order


    l_result := DBMS_PIPE.SEND_MESSAGE(
                    pipename => 'ORDER_PIPE',
                    credential_name => DBMS_PIPE.GET_CREDENTIAL_NAME,
```

```
                        location_uri => DBMS_PIPE.GET_LOCATION_URI);

    IF l_result = 0 THEN
        DBMS_OUTPUT.put_line('DBMS_PIPE sent order successfully');
    END IF;

END;
/
```

See PACK_MESSAGE Procedures and SEND_MESSAGE Function for more information.

Retrieve a Persistent Message on Same Database

Describes the steps to retrieve a persistent message from an explicit pipe on the same Autonomous Database instance (the instance where the message was sent).

On an Autonomous Database instance you can receive messages sent to a pipe from a different session. The `DBMS_PIPE` procedures are invoker's rights procedures and run as the current invoked user.

Private pipes are owned by the current user that creates the pipe. Private pipes can only be accessed by the same user that created the pipe. This applies to pipes using in-memory messages and to pipes using persistent messaging with messages stored in Cloud Object Store.

Public pipes can be accessed by any database session having execute privilege on `DBMS_PIPE`. This applies to pipes using in-memory messages and to pipes using persistent messaging with messages stored in Cloud Object Store.

1. Verify that the pipe is created.

```
SELECT ownerid, name, type FROM v$db_pipes
       WHERE name = 'ORDER_PIPE';


OWNERID NAME       TYPE
------- ---------- -------
     80 ORDER_PIPE PRIVATE
```

When you are on the same Autonomous Database instance and the pipe exists, you do not need to run `DBMS_PIPE.CREATE_PIPE` before you receive a message. This applies when the pipe was created on the same instance, as shown in Create an Explicit Persistent Pipe and Send a Message.

2. Receive a message from the pipe.

```
DECLARE
    message1  DATE;
    message2  VARCHAR2(100);
    message3  INTEGER;
    message4  VARCHAR2(100);
    l_result  INTEGER;

BEGIN

    DBMS_PIPE.SET_CREDENTIAL_NAME('my_persistent_pipe_cred');
    DBMS_PIPE.SET_LOCATION_URI('https://objectstorage.us-
```

```
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/');
    l_result := DBMS_PIPE.RECEIVE_MESSAGE (
                  pipename => 'ORDER_PIPE',
                  timeout  => DBMS_PIPE.MAXWAIT,
                  credential_name => DBMS_PIPE.GET_CREDENTIAL_NAME,
                  location_uri => DBMS_PIPE.GET_LOCATION_URI);

    IF l_result = 0 THEN
        DBMS_PIPE.unpack_message(message1);
        DBMS_PIPE.unpack_message(message2);
        DBMS_PIPE.unpack_message(message3);
        DBMS_PIPE.unpack_message(message4);

        DBMS_OUTPUT.put_line('Order Received Successfully On: ' ||
TO_CHAR(sysdate, 'dd-mm-yyyy hh24:mi:ss'));
        DBMS_OUTPUT.put_line('Date of Order: ' || message1);
        DBMS_OUTPUT.put_line('Order Number: ' || message2);
        DBMS_OUTPUT.put_line('Number of Items In Order: ' || message3);
        DBMS_OUTPUT.put_line('Item Type in Order: ' || message4);
    END IF;

END;
/
```

When you are on the same Autonomous Database instance, the credential already exists and you do not need to run `DBMS_CLOUD.CREATE_CREDENTIAL` to receive a message. This applies when the pipe was created on the same instance, as shown in Create an Explicit Persistent Pipe and Send a Message.

See SET_CREDENTIAL_NAME Procedure and SET_LOCATION_URI Procedure for more information.

See RECEIVE_MESSAGE Function for more information.

Retrieve a Persistent Message by Creating a Pipe on a Different Database

Describes the steps to retrieve a persistent message stored in Cloud Object Store with an explicit pipe on an Autonomous Database instance that is different than the instance that sent the message.

1. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`. For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'my_persistent_pipe_cred',
    username => 'adb_user@example.com',
    password => 'password'
  );
END;
/
```

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same

credential name to access the Cloud Object Store to send and receive messages with DBMS_PIPE.

For detailed information about the parameters, see CREATE_CREDENTIAL Procedure.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

> **Note:**
>
> Some tools like SQL*Plus and SQL Developer use the ampersand character (&) as a special character. If you have the ampersand character in your password use the SET DEFINE OFF command in those tools as shown in the example to disable the special character and get the credential created properly.

2. Create an explicit pipe with the same name as the pipe that sent the message. For example, create a pipe named ORDER_PIPE.

```
DECLARE
  r_status INTEGER;
BEGIN
    r_status := DBMS_PIPE.CREATE_PIPE(pipename => 'ORDER_PIPE');
END;
/
```

See CREATE_PIPE Function.

3. Verify that the pipe is created.

```
SELECT ownerid, name, type FROM v$db_pipes
       WHERE name = 'ORDER_PIPE';


OWNERID NAME       TYPE
------- ---------- -------
     80 ORDER_PIPE PRIVATE
```

4. Use DBMS_PIPE procedures to set the default access credential and location URI for Object Store so that DBMS_PIPE can access the persistent message.

```
BEGIN
    DBMS_PIPE.SET_CREDENTIAL_NAME('my_persistent_pipe_cred');
    DBMS_PIPE.SET_LOCATION_URI('https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/');
END;
/
```

These procedures set the default credential name and the default location URI for use with DBMS_PIPE procedures.

If you use Oracle Cloud Infrastructure Object Storage to store messages, you can use Oracle Cloud Infrastructure Native URIs or Swift URIs. However, the location URI and the credential must match in type as follows:

- If you use a native URI format to access Oracle Cloud Infrastructure Object Storage, you must use Native Oracle Cloud Infrastructure Signing Keys authentication in the credential object.
- If you use Swift URI format to access Oracle Cloud Infrastructure Object Storage, you must use an auth token authentication in the credential object.

See SET_CREDENTIAL_NAME Procedure and SET_LOCATION_URI Procedure for more information.

5. Receive a message from the persistent pipe.

```
DECLARE
    message1  DATE;
    message2  VARCHAR2(100);
    message3  INTEGER;
    message4  VARCHAR2(100);
    l_result  INTEGER;

BEGIN

    DBMS_PIPE.SET_CREDENTIAL_NAME('my_persistent_pipe_cred');
    DBMS_PIPE.SET_LOCATION_URI('https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/');
    l_result := DBMS_PIPE.RECEIVE_MESSAGE (
                  pipename => 'ORDER_PIPE',
                  timeout  => DBMS_PIPE.MAXWAIT,
                  credential_name => DBMS_PIPE.GET_CREDENTIAL_NAME,
                  location_uri => DBMS_PIPE.GET_LOCATION_URI);

    IF l_result = 0 THEN
        DBMS_PIPE.unpack_message(message1);
        DBMS_PIPE.unpack_message(message2);
        DBMS_PIPE.unpack_message(message3);
        DBMS_PIPE.unpack_message(message4);

        DBMS_OUTPUT.put_line('Order Received Successfully On: ' ||
TO_CHAR(sysdate, 'dd-mm-yyyy hh24:mi:ss'));
        DBMS_OUTPUT.put_line('Date of Order: ' || message1);
        DBMS_OUTPUT.put_line('Order Number: ' || message2);
        DBMS_OUTPUT.put_line('Number of Items In Order: ' || message3);
        DBMS_OUTPUT.put_line('Item Type in Order: ' || message4);
    END IF;

END;
/
```

See RECEIVE_MESSAGE Function for more information.

Remove a Persistent Pipe

Describes the steps to remove a persistent pipe.

Persistent pipes send and receive messages by storing messages in Cloud Object Store. Use `DBMS_PIPE.REMOVE_PIPE` to remove a persistent pipe on an Autonomous Database instance.

1. Call the DBMS_PIPE.REMOVE_PIPE function to remove a pipe.

```
DECLARE
   l_result  INTEGER;
BEGIN
     l_result := DBMS_PIPE.REMOVE_PIPE('ORDER_PIPE');
END;
/
```

   The REMOVE_PIPE function removes the pipe from the Autonomous Database instance where it runs, however REMOVE_PIPE does not affect other Autonomous Database instances with a pipe with the same name that uses the same location URI.

2. On the Autonomous Database instance were you run DBMS_PIPE.REMOVE_PIPE, verify that the pipe is removed.

```
SELECT ownerid, name, type FROM v$db_pipes
       WHERE name = 'ORDER_PIPE';


No rows selected
```

Query Text in Object Storage

The PL/SQL package DBMS_CLOUD enables you to build a text index on the object store files, which allows you to search the text and use wildcards with your search.

- About Using a Text Index to Query Text on Object Storage
- Create a Text Index on Object Storage Files
  Use DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX to create a text index on files in object storage.
- Drop an Index on the Cloud Storage Files
  Use the DBMS_CLOUD.DROP_EXTERNAL_TEXT_INDEX procedure to drop a text index on object storage files.
- Text Index Reference Table
  A local table is created within your database with a standard suffix *INDEX_NAME*$TXTIDX. This table is created internally when you run DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX.
- Monitor Text Index Creation
  When you run DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX the text index creation operation is logged in the ALL_SCHEDULER_JOB_RUN_DETAILS view.

About Using a Text Index to Query Text on Object Storage

You can create a text index on files in object storage. A text index allows you to perform a word-based search across very large data sets in object store.

DBMS_CLOUD provides fast and efficient ways to manage data in object store. The DBMS_CLOUD APIs let you create, copy, download, delete, and traverse files present in object store. When you define external tables you can run SQL queries on data stored in your object store (or with hybrid partitioned external tables, across data in your database and in object store). When you use DBMS_CLOUD to define a text index, this allows you to search your data for text and use wildcards.

Autonomous Database support for word-based search works for commonly used data formats, for example CSV or JSON and with formatted documents (binary), for example PDF and DOC (MS Word) formats. You can configure a refresh rate that indicates the frequency in minutes at which the index is refreshed for any new uploads or deletes.

A local table with the standard suffix *INDEX_NAME*$TXTIDX is created when you create an index on the object storage, and you can utilize the table to perform a search using the CONTAINS keyword.

See Indexing with Oracle Text for more information.

Create a Text Index on Object Storage Files

Use DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX to create a text index on files in object storage.

Formatted documents (binary) are supported when you specify the binary_files format option with DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX.

You can include a stop word list when you specify the stop_words format option with DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX.

See Indexing with Oracle Text for more information on Oracle Text stop words and working with binary files.

1. Create a credential object to access the source location.

   See CREATE_CREDENTIAL Procedure for more information.

2. Run the DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX procedure to create a text index on the object storage files.

```
BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX (
        credential_name => 'DEFAULT_CREDENTIAL',
        location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/ts_data/'
        index_name      => 'EMP',
        format          => JSON_OBJECT ('refresh_rate' value 10)
);
END;
/
```

   This example creates a text index EMP on the object storage files located at the URI specified in the location_uri parameter. The refresh_rate option in the format parameter specifies that the index EMP is refreshed at an interval of 10 minutes.

   This creates a local table *INDEX_NAME*$TXTIDX. You can utilize the table *INDEX_NAME*$TXTIDX to perform a search using CONTAINS.

   For example:

```
SELECT object_name FROM EMP$TXTIDX
     WHERE CONTAINS(object_name,'king') > 0;
```

   This query returns the object or file names that contain the string king.

   See Text Index Reference Table for more information.

You can query an external table using the `EXTERNAL MODIFY` clause to retrieve the actual records.

```
SELECT * FROM EMPEXTTAB EXTERNAL MODIFY ((location_url object_name));
```

> **Note:**
>
> The external table `EMPEXTTAB` is a sample external table that is created on the same `location_url`.

See Querying External Data with Autonomous Database for more information.

See CREATE_EXTERNAL_TEXT_INDEX Procedure for more information.

See Configure Policies and Roles to Access Resources for more information.

Drop an Index on the Cloud Storage Files

Use the `DBMS_CLOUD.DROP_EXTERNAL_TEXT_INDEX` procedure to drop a text index on object storage files.

Run the `DBMS_CLOUD.DROP_EXTERNAL_TEXT_INDEX` procedure to drop a text index on files in object storage.

```
BEGIN
DBMS_CLOUD.DROP_EXTERNAL_TEXT_INDEX (
        index_name => 'EMP',
);
END;
/
```

This example drops the `EMP` text index.

See DROP_EXTERNAL_TEXT_INDEX Procedure for more information.

Text Index Reference Table

A local table is created within your database with a standard suffix *`INDEX_NAME`*`$TXTIDX`. This table is created internally when you run `DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX`.

You can query the *`INDEX_NAME`*`$TXTIDX` table to search for a string using the `CONTAINS` keyword. For example, when you call `DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX` procedure the *`INDEX_NAME`* value as `EMP`, this creates the `EMP$TXTIDX` the text reference table.

The text reference table has the following columns:

- `object_name`: is the file name on the object storage that contains the searched text string.
- `object_path`: is the object storage bucket or folder URI that contains the object storage file.
- `mtime`: is the last modified timestamp of the object storage file. This is the time when the file was last accessed by `DBMS_CLOUD`.

For example:

```
SELECT object_path, object_name FROM EMP$TXTIDX WHERE CONTAINS(OBJECT_NAME, 'king') > 0;

OBJECT_PATH
    OBJECT_NAME
--------------------------------------------------------------------------------
   ----------------------------------
https://objectstorage.us-phoenix-1.oraclecloud.com/n/example1/b/adbs_data_share/o/
ts_data/       data_2_20221026T195313585601Z.json
```

This query returns the file names and location URI on the object storage which contains the text string `king`, in either upper or lowercase.

```
SELECT object_name, mtime FROM EMP$TXTIDX;

OBJECT_NAME                    MTIME
------------------------------ ------------------------------------
data_1_20220531T165402Z.json    31-MAY-22 04.54.02.979000 PM +00:00
data_1_20220531T165427Z.json    31-MAY-22 04.54.27.997000 PM +00:00
```

This query returns file name and last modified timestamp of the object files on which the index `EMP` is created.

Monitor Text Index Creation

When you run `DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX` the text index creation operation is logged in the `ALL_SCHEDULER_JOB_RUN_DETAILS` view.

You can query the `ALL_SCHEDULER_JOB_RUN_DETAILS` view to obtain the status and any error reported by the index creation job.

The name of the `DBMS_SCHEDULER` job is derived from the `INDEX_NAME` parameter specified when you call `DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX`.

To query the `ALL_SCHEDULER_JOB_RUN_DETAILS` view, you must be logged in as the `ADMIN` user or have `READ` privilege on the `ALL_SCHEDULER_JOB_RUN_DETAILS` view.

For example, the following `SELECT` statement with a `WHERE` clause on `job_name` shows the run details for the job:

```
SELECT status, additional_info
   FROM all_scheduler_job_run_details WHERE LOWER(job_name) = LOWER('index_name$JOB');
```

You can also query for the existence of an index creation scheduler job.

For example:

```
SELECT status
    FROM all_scheduler_jobs where LOWER(job_name) = LOWER('index_name$JOB');
```

See CREATE_EXTERNAL_TEXT_INDEX Procedure for more information.

Use Oracle Workspace Manager on Autonomous Database

Workspace Manager provides an infrastructure that enables applications to create workspaces and group different versions of table row values in different workspaces.

- About Using Oracle Workspace Manager on Autonomous Database
  Use Oracle Workspace Manager to version-enable one or more user tables in the database. When a table is version-enabled, all rows in the table can support multiple versions of the data.
- Enable Oracle Workspace Manager on Autonomous Database
  Oracle Workspace Manager must be enabled to be used on Autonomous Database. You can migrate existing data that has enabled Oracle Workspace Manager.
- Disable Oracle Workspace Manager on Autonomous Database
  Use the following steps to disable Oracle Workspace Manager on Autonomous Database.

About Using Oracle Workspace Manager on Autonomous Database

Use Oracle Workspace Manager to version-enable one or more user tables in the database. When a table is version-enabled, all rows in the table can support multiple versions of the data.

Applications that can benefit from Workspace Manager typically do one or more of the following operations:

- Manage a collection of updates and insertions as a unit before incorporating them into production data
- Support a collaborative development effort
- Use a common data set to create multiple scenarios for what-if analyses or multiple editions of data for publication
- Keep a history of changes to data

Introduction to Workspace Manager provides relevant usage and reference information.

Workspace manager is enabled using the procedure `DBMS_CLOUD_ADMIN.ENABLE_FEATURE`.

Enable Oracle Workspace Manager on Autonomous Database

Oracle Workspace Manager must be enabled to be used on Autonomous Database. You can migrate existing data that has enabled Oracle Workspace Manager.

As the `ADMIN` user, run `DBMS_CLOUD_ADMIN.ENABLE_FEATURE` to enable Oracle Workspace Manager.

1. Enable Oracle Workspace Manager.

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_FEATURE(
        feature_name => 'OWM');
END;
/
```

2. Restart the Autonomous Database instance.

   See Restart Autonomous Database for more information.

3. Query `dba_cloud_config` to verify that Oracle Workspace Manager is enabled.

```
SELECT param_name, param_value FROM dba_cloud_config WHERE
       UPPER(param_name) = 'OWM';

PARAM_NAME PARAM_VALUE
---------- -----------
owm        enabled
```

4. (Optional) If you have existing data using Oracle Workspace Manager on another database that you want to migrate to Autonomous Database, Oracle Workspace Manager provides import and export procedures that enable you to migrate the data. See Export_Schemas and Import_Schemas for additional information.
5. Continue with the imported schemas or the new schemas.

See ENABLE_FEATURE Procedure for more information.

Disable Oracle Workspace Manager on Autonomous Database

Use the following steps to disable Oracle Workspace Manager on Autonomous Database.

As the `ADMIN` user, run `DBMS_CLOUD_ADMIN.DISABLE_FEATURE` to disable Oracle Workspace Manager.

1. Disable Oracle Workspace Manager.

```
BEGIN
   DBMS_CLOUD_ADMIN.DISABLE_FEATURE(
        feature_name => 'OWM');
END;
/
```

2. Query `dba_cloud_config` to verify that Oracle Workspace Manager is disabled.

```
SELECT param_name, param_value FROM dba_cloud_config
     WHERE UPPER(param_name) = 'OWM';

0 rows selected.
```

See DISABLE_FEATURE Procedure for more information.

Use and Manage Elastic Pools on Autonomous Database

Use an elastic pool to consolidate your Autonomous Database instances, in terms of their allocation of compute resources, and to provide up to 87% cost savings.

Elastic pools help you improve operating efficiency and reduce costs by bringing all of your databases to the Cloud. This also supports consolidating resources and simplifying administration and operations by using Autonomous Database. When you need a large number of databases, that can scale up and down elastically without downtime, you can benefit by creating and using elastic pools.

Elastic pools have the following advantages:

- Enable operating with a fixed budget for a group of databases, while delivering performance elasticity for each individual database.

- Allow for easy migration from on-prem Oracle environments that include oversubscription, to provide a cost effective way to move to Autonomous Database.
- Support SaaS vendors with a large number of individual customer databases.
- Provide resources for using a microservices architecture, where the ability to supply of large number of databases is required.
- The pool members in an elastic pool are not billed individually (the pool leader is billed based on the pool shape). You can allocate additional ECPUs per instance for pool members, without worrying about the cost associated with the ECPU usage for the individual members. Autonomous Database IO capacity and memory allocation is directly correlated with the **ECPU count**, so by selecting a greater number of ECPUs for an instance, this allows you to run with greater IO capacity and more memory without having to pay for the additional resources. This means, using a larger number of ECPUs per instance allows you to use more IO capacity and more memory per instance, where the cost is based on the pool shape and is not based on an individual instance's **ECPU count**.

> **Note:**
>
> Elastic pools are only available for Autonomous Database instances that use the ECPU compute model.

Topics

About Elastic Pools

Use an elastic pool to consolidate your Autonomous Database instances, in terms of their allocation of compute resources, and to provide up to 87% cost savings.

When you create an elastic pool you select a pool size from a predefined set of pool sizes. Pool size determines how much you pay for compute as well as how many ECPUs you can provision in a given pool.

There are several terms to use when you work with elastic pools:

- **Pool Leader**: Is the Autonomous Database instance that creates an elastic pool.
- **Pool Member**: Is an Autonomous Database instance that is added to an elastic pool.
- **Pool Size**: Is a value that you set when you create an elastic pool. The pool size must be one of the available elastic pool shapes.
- **Pool Shape**: A pool shape is one of the valid pool sizes that you select when you create an elastic pool. The pool shape must be one of: 128, 256, 512, 1024, 2048, or 4096 ECPUs.

> **Note:**
>
> By default each instance in an elastic pool is automatically assigned a maintenance window. By selecting a pool shape that is 1024 ECPUs or greater, you have the option of assigning a custom 2-hour maintenance window during which the leader and all elastic pool members are patched together. To select a custom maintenance window for your elastic pool, file a Service Request at Oracle Cloud Support.

- **Pool Capacity**: The pool capacity is the maximum number of ECPUs that an elastic pool can use, and is four times (x4) the pool size.

Requirements to Create an Elastic Pool

The following are the requirements for an Autonomous Database instance to create an elastic pool and become a pool leader:

- The instance must use the ECPU compute model.
- The instance must be an Autonomous Database instance with the **Transaction Processing** workload type. This only applies for the pool leader. An elastic pool can hold a mix of databases with **Transaction Processing**, **Data Warehouse**, **JSON Database**, or **APEX** workloads.
- Auto scaling must be disabled.
- The instance must not be a member of an existing elastic pool.
- The maximum allowed individual **ECPU count** for an Autonomous Database instance that creates an elastic pool is 4 times the pool size specified when you create the pool.
- The instance that creates an elastic pool is subject to tenancy limits. To create an elastic pool you must have a sufficient number of ECPUs available, below the tenancy limit, to accommodate the size of the elastic pool.

Requirements to Join an Elastic Pool

The following are the requirements for an Autonomous Database instance to join an elastic pool:

- The instance must use the ECPU compute model.
- An elastic pool can contain Autonomous Database instances with **Transaction Processing**, **Data Warehouse**, **JSON Database**, or **APEX** workload types.
- An elastic pool can hold a mix of databases with **Transaction Processing**, **Data Warehouse**, **JSON Database**, and **APEX** workloads.
- Auto scaling must be disabled.
- The instance must not be a member of an elastic pool.

- The maximum allowed individual **ECPU count** for an Autonomous Database instance is the available pool capacity. When an instance has an ECPU count greater than the available pool capacity, it is not allowed to join that elastic pool.

Pool Leader and Member Instance ECPU Allocation

When an Autonomous Database instance is part of an elastic pool, the minimum allowed individual ECPU allocation for an instance is 1 ECPU.

When an Autonomous Database instance is part of an elastic pool, increments of 1 ECPU are allowed for individual Autonomous Database instance ECPU allocation.

Pool Capacity for an Elastic Pool

An elastic pool has a pool capacity of 4 times the pool size. For example, a pool with pool size of 128 ECPUs can hold up to 512 ECPUs for its leader and the members.

> **Note:**
>
> In these examples Autonomous Data Guard is not enabled. See About Elastic Pools with Autonomous Data Guard Enabled for information on using elastic pools with Autonomous Data Guard.

The following are examples of Autonomous Database instances that could be in an elastic pool with a pool size of 128 and a pool capacity of 512 ECPUs:

- Each of these are valid for pool members in an elastic pool with a pool size of 128 ECPUs:
  - 1 instance with 512 ECPUs, for a total of 512 ECPUs
  - 128 instances with 4 ECPUs, for a total of 512 ECPUs
  - 256 instances with 2 ECPUs, for a total of 512 ECPUs
- Similarly, each of the following are valid for pool members in an elastic pool with a pool size of 128 ECPUs:
  - 1 instance with 128 ECPUs, 2 instances with 64 ECPUs, 32 instances with 4 ECPUs, and 64 instances with 2 ECPUs, for a total of 512 ECPUs
  - 256 instances with 1 ECPU, 64 instances with 2 ECPUs, for a total of 384 ECPUs, which is less than the pool capacity of 512 ECPUs.

Topics

About Elastic Pool Billing

Elastic pool usage is billed to the pool leader and billing is based on the elastic pool size and the actual hourly ECPU usage of the pool leader and the members. Elastic pool usage can exceed the pool size (pool capacity can be up to four times greater than the pool size).
The billing for an elastic pool consists of only compute resources, that is ECPU usage, and all compute usage is charged to the Autonomous Database instance that is the pool leader. Any billing for storage usage is charged separately to individual Autonomous Database instances, independent of whether the instance is in an elastic pool.

An elastic pool allows you to consolidate your Autonomous Database instances in terms of their compute resource billing. You can think of an elastic pool like a mobile phone service “family plan”, except this applies to your Autonomous Database instances. Instead of paying individually for each database, the databases are grouped into a pool in which one instance, the leader, is charged for the compute usage associated with the entire pool.

Using an elastic pool you are able to provision up to four times the number of ECPUs, over your selected pool size, and you can provision database instances that are in the elastic pool with as little as 1 ECPU per database instance. Outside of an elastic pool the minimum number of ECPUs per database instance is 2 ECPUs. For example, with a pool size of 128 you can provision 512 Autonomous Database instances (when each instance has 1 ECPU). In this example you are billed for the pool size compute resources, based on the pool size of 128 ECPUs, while you have access to 512 Autonomous Database instances. In contrast, when you individually provision 512 Autonomous Database instances without using an elastic pool you are required to allocate a minimum of 2 ECPUs for each Autonomous Database instance, and in this example you would pay for 1024 ECPUs. Using an elastic pool provides **up to 87% compute cost savings**.

After you create an elastic pool, the total ECPU usage for a given hour is charged to the Autonomous Database instance that is the pool leader. With the exception of the pool leader, individual Autonomous Database instances that are pool members are not charged for ECPU usage while they are members of an elastic pool.

Elastic pool billing is as follows:

- If the total aggregated peak ECPU utilization is equal to or below the pool size for a given hour, you are charged for the pool size number of ECPUs (one times the pool size).

  After an elastic pool is created ECPU billing continues at a minimum of one times the pool size rate, even when databases that are part of the pool are stopped. This applies to pool member databases and to the pool leader.

  In other words, if the aggregated peak ECPU utilization of the pool is less than or equal to the pool size for a given hour, you are charged for the pool size number of ECPUs (one times the pool size). This represents **up to 87% compute cost savings** over the case in which these databases are billed separately without using elastic pools.

- If the aggregated peak ECPU utilization of the pool leader and the members exceeds the pool size at any point in time in a given billing hour:

  – **Aggregated peak ECPU utilization of the pool is equal to or less than two times of the pool size number of ECPUs**: For usage that is greater than one times the pool size number of ECPUs and up to and including two times the number of ECPUs in a given billing hour: Hourly billing is two times the pool size number of ECPUs.

    In other words, if the aggregated peak ECPU utilization of the pool exceeds the pool size, but is less than or equal to two times the pool size for a given hour, you are charged for twice the pool size number of ECPUs (two times the pool size). This

represents **up to 75% compute cost savings** over the case in which these databases are billed separately without using elastic pools.

- **Aggregated peak ECPU utilization of the pool is equal to or less than four times the pool size number of ECPUs**: For usage that is greater than two times the pool size number of ECPUs and up and including to four times the pool size number of ECPUs in a given billing hour: Hourly billing is four times the pool size number of ECPUs.

  In other words, if the aggregated peak ECPU utilization of the pool exceeds twice the pool size for a given hour, you are charged for four times the pool size number of ECPUs (four times the pool size). This represents **up to 50% compute cost savings** over the case in which these databases are billed separately without using elastic pools.

For example, consider an elastic pool with a pool size of 128 ECPUs and a pool capacity of 512 ECPUs:

- **Case-1**: The aggregated peak ECPU utilization of the pool leader and the members is 40 ECPUs between 2:00pm and 2:30pm, and 128 ECPUs between 2:30pm and 3:00pm.

  The elastic pool is billed 128 ECPUs, one times the pool size, for this billing hour (2-3pm). This case applies when the peak aggregated ECPU usage of the elastic pool for the billing hour is less than or equal to 128 ECPUs.

- **Case-2**: The aggregated peak ECPU utilization of the pool leader and the members is 40 ECPUs between 2:00pm and 2:30pm, and 250 ECPUs between 2:30pm and 3:00pm.

  The elastic pool is billed 256 ECPUs, two times the pool size, for this billing hour (2-3pm). This case applies when the peak aggregated ECPU usage of the elastic pool for the billing hour is less than or equal to 256 ECPUs and greater than 128 ECPUs.

- **Case-3**: The aggregated peak ECPU utilization of the pool leader and the members is 80 ECPUs between 2:00pm and 2:30pm, and 509 ECPUs between 2:30pm and 3:00pm.

  The elastic pool is billed 512 ECPUs, four times the pool size, for this billing hour (2-3pm). This case applies when the peak aggregated ECPU usage of the elastic pool for the billing hour is less than or equal to 512 ECPUs and greater than 256 ECPUs.

See How to Achieve up to 87% Compute Cost Savings with Elastic Resource Pools on Autonomous Database for more details.

Elastic Pool Billing when a Pool is Created or Terminated

When an elastic pool is created or terminated, the leader is billed for the full hour for the elastic pool. In addition, individual instances that are either added or removed from the pool are billed for any compute usage that occurs while the instance is not in the elastic pool (in this case the billing applies to the individual Autonomous Database instance).

- **Pool Creation Example**: Assume there is an Autonomous Database instance with 4 ECPUs that is not part of any elastic pool. At 2:15pm, if you create an elastic pool with this instance with a pool size of 128 ECPUs, the instance becomes a pool leader. Assuming the Autonomous Database idles between 2-3pm, and there are no other Autonomous Database instances in the pool, billing for the hour between 2-3pm is as follows:

  The bill for the period 2-3pm is: (4 * 0.25) + 128 = 129 ECPUs

  Where the (4 * 0.25) is the billing for compute for the fifteen minutes before the Autonomous Database instance created the elastic pool, and 128 ECPUs is the billing for the elastic pool for the hour when the elastic pool is created.

- **Pool Termination Example**: Assume an Autonomous Database instance with 4 ECPUs is the leader of an elastic pool and the pool size is 128 ECPUs. At 4:30pm, if you terminate the elastic pool, the database becomes a standalone Autonomous Database instance that is not part of any elastic pool. Assuming the Autonomous Database idles between 4-5pm, and there are no other Autonomous Database instances in the pool, billing for the hour between 4-5pm is as follows:

  The bill for 4-5pm is: (4 * 0.5) + 128 = 130 ECPUs

  Where the (4 * 0.5) is the billing for compute for the thirty minutes after the Autonomous Database instance terminates the elastic pool, and 128 ECPUs is the billing for the elastic pool for the hour when the elastic pool was terminated.

Elastic Pool Billing with Built-in Tools

For either the pool leader or the members, compute resources that are allocated to the built-in tools, OML, Graph, or Data Transforms, are separate and do not count towards the elastic pool total allocation. For billing purposes, the elastic pool leader is billed for any built-in tool ECPU usage by either the leader or elastic pool members, in addition to the elastic pool ECPU usage.

For example, assume there is an elastic pool with a pool size of 128 ECPUs; if in a given billing hour the aggregated peak ECPU utilization of the pool leader and the members is 80 ECPUs for the billing hour, and during this hour the combined total ECPU utilization for instances using built-in tools is 30 ECPUs, the leader is charged for the pool size (128 ECPUs), plus the built-in tool ECPU usage (30 ECPUs), for a total of 158 ECPUs for that hour.

About Elastic Pools with Autonomous Data Guard Enabled

The elastic pool leader or members can enable either local or cross-region Autonomous Data Guard, or both local and cross-region Autonomous Data Guard.

Local Autonomous Data Guard Standby Database Billing

When you add a local standby, a total of 2x the primary's ECPU allocation is counted towards the pool capacity (1x for the primary and 1x for the standby).

For example, if you create an elastic pool with a pool size of 128 ECPUs, with a pool capacity of 512 ECPUs, adding the following Autonomous Database instance uses the entire elastic pool capacity:

- 1 instance with 256 ECPUs with local Autonomous Data Guard enabled, for a total of 512 ECPUs allocation from the pool.

Similarly, if you create an elastic pool with a pool size of 128 ECPUs, with a pool capacity of 512 ECPUs, adding the following Autonomous Database instances uses the entire elastic pool capacity:

- 128 instances with 2 ECPUs each with local Autonomous Data Guard enabled, for a total of 512 ECPUs allocation from the pool.

Cross-Region Autonomous Data Guard Standby Database Billing

Enabling Cross-region Autonomous Data Guard for a leader or for member has no effect on the elastic pool capacity. A cross-region Autonomous Data Guard peer database has its own OCID and the cross-region peer is billed independently from the elastic pool.

Note the following:

- Cross-region Autonomous Data Guard peer ECPUs do not use pool capacity and billing for Autonomous Data Guard cross-region peer databases happens on the peer instance.

- When the leader of an elastic pool enables cross-region Autonomous Data Guard, the cross-region peer database ECPU allocation does not count towards the elastic pool capacity. Billing for cross-region Autonomous Data Guard is on the cross-region instance, which is not part of the elastic pool (elastic pools do not operate across regions).
- When a member of an elastic pool enables cross-region Autonomous Data Guard, the cross-region peer ECPU allocation does not count towards the pool capacity. Billing for cross-region Autonomous Data Guard is on the cross-region instance, which is not part of the elastic pool (elastic pools do not operate across regions).

For example, if you create an elastic pool with a pool size of 128 ECPUs (with a pool capacity of 512 ECPUs), adding the following Autonomous Database instances of different sizes uses the entire elastic pool capacity:

- A pool that contains the following instances:
  - 1 instance with 128 ECPUs with cross-region Autonomous Data Guard enabled (using a total of 128 ECPUs from the pool).
  - 64 instances with 2 ECPUs each with both local and cross-region Autonomous Data Guard enabled (using a total of 256 ECPUs from the pool).
  - 128 instances with 1 ECPU, each with cross-region Autonomous Data Guard enabled (using 128 ECPUs from the pool).

About Elastic Pool Leader and Member Operations

The Autonomous Database instance that creates an elastic pool is the pool leader. Autonomous Database instances that are added to an existing pool are pool members. Depending on your role, either leader or member, you can perform operations on an elastic pool.

Pool Leader Operations

The following operations are valid only for the pool leader:

Operation	Description
Create an elastic pool	The Autonomous Database instance that creates an elastic pool is the pool leader. See Create Join or Manage an Elastic Pool for more information.
Remove an elastic pool member	An elastic pool leader can remove a member from the elastic pool. See As Pool Leader Remove Members from an Elastic Pool for more information.
Terminate an elastic pool	When an elastic pool has no pool members, the pool leader can terminate the elastic pool. See Terminate an Elastic Pool for more information.
Modify elastic pool size	An elastic pool leader can modify the pool size. See Change the Elastic Pool Shape for more information.
List pool members	A pool leader can list pool members. See List Elastic Pool Members for more information.

Pool Member Operations

The following operations are valid for a pool member or for the pool leader:

Operation	Description
Add instance to elastic pool	An Autonomous Database instance can be added as a pool member as long as the instance is one of the supported workload types, the instance uses the ECPU compute model, and the instance is not a pool member of a different pool. The supported workload types are: **Transaction Processing**, **Data Warehouse**, **JSON Database**, or **APEX**. See Join an Existing Elastic Pool for more information.
Remove an elastic pool member	An elastic pool member can remove themselves from the elastic pool. See Remove Pool Members from an Elastic Pool for more information.

Create Join or Manage an Elastic Pool

Shows the steps to create, join, or change the pool size for an elastic pool.

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then click Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

To create an elastic pool:

> **Note:**
>
> To create an elastic pool the instance must use the ECPU compute model and the workload type must be **Transaction Processing**.

1. On the Autonomous Database Details page select **Manage resource allocation**.

   Verify that **Compute auto scaling** is disabled.

   If **Compute auto scaling** is selected, disable **Compute auto scaling**:

   a. In the **Manage resource allocation** area, deselect **Compute auto scaling**.

   b. Click **Apply**.

      The **Lifecycle state** changes to **Scaling in Progress**. After some time the **Lifecycle state** changes to **Available**.

   c. On the Autonomous Database Details page select **Manage resource allocation** to display the **Manage resource allocation** area.
2. In the **Manage resource allocation** area, click **Show advanced options**.
3. Select **Enable elastic pool**.
4. Select **Create a elastic pool**.
5. Enter or choose a pool size in the **Pool ECPU count** field.

   Select a pool size from the list of pool shapes: 128, 256, 512, 1024, 2048, or 4096.

By default each instance in an elastic pool is automatically assigned a maintenance window. By selecting a **Pool ECPU count** that is 1024 greater, you have the option of assigning a custom 2-hour maintenance window during which the leader and all elastic pool members are patched together. To select a custom maintenance window for your elastic pool, file a Service Request at Oracle Cloud Support.

See About Scheduled Maintenance and Patching for more information.

6. Click **Apply** to create the elastic pool.

When you click **Apply**, the **Lifecycle state** changes to **Scaling in Progress**. After the **Lifecycle state** changes to **Available** the changes apply immediately.

After you create an elastic pool, click **Manage resource allocation** to display the elastic pool information. In the **Manage resource allocation** area, the **Elastic pool** field shows **Enabled**, the **Pool role** field shows **Leader**, and the **Pool ECPU count** field shows the pool size you selected.

- Join an Existing Elastic Pool
  Shows the steps to join an existing elastic pool.
- Change the Elastic Pool Shape
  Shows the steps for the pool leader to change the elastic pool shape for an existing elastic pool.

Join an Existing Elastic Pool

Shows the steps to join an existing elastic pool.

To join an elastic pool, the following is required for the Autonomous Database instance:

- The instance must use the ECPU compute model.

  See Compute Models in Autonomous Database for more information.
- The workload type must be one of **Transaction Processing**, **Data Warehouse**, **JSON Database**, or **APEX**.

  See About Autonomous Database Workload Types for more information.
- Auto scaling must be disabled.
- The instance must not be a member of an elastic pool.

To join an elastic pool:

1. On the Autonomous Database Details page select **Manage resource allocation**.

   Verify that **Compute auto scaling** is disabled.

   If **Compute auto scaling** is selected, disable **Compute auto scaling**:

   a. In the **Manage resource allocation** area, deselect **Compute auto scaling**.

   b. Click **Apply**.

      The **Lifecycle state** changes to **Scaling in Progress**. After some time the **Lifecycle state** changes to **Available**.

   c. On the Autonomous Database Details page select **Manage resource allocation** to display the **Manage resource allocation** area.

2. In the **Manage resource allocation** area, click **Show advanced options** to show the advanced options.
3. Select **Enable elastic pool**.
4. Select **Join an existing elastic pool**.
5. In the **Select pool leader in *compartment*** field choose a pool leader in a compartment.

   a. Use the compartment shown or click **Change Compartment** to select a different compartment.

   b. Select a pool leader from the list of available pool leaders in the selected compartment.

6. Click **Apply** to add the instance to the elastic pool.

When you click **Apply** the **Lifecycle state** changes to **Scaling in Progress**. After the **Lifecycle state** changes to **Available** the changes apply immediately.

After you create an elastic pool, click **Manage resource allocation** to see the elastic pool details. In the Manage resource allocations area, under **Elastic pool**, the **Elastic pool** field shows **Enabled**, the **Pool role** field shows **Member**, and the **Elastic pool leader** field shows a link to the pool leader.

Change the Elastic Pool Shape

Shows the steps for the pool leader to change the elastic pool shape for an existing elastic pool.

> **Note:**
>
> Only a pool leader can modify the pool shape.

To change the shape of an elastic pool (update the pool size):

1. On the Autonomous Database Details page select **Manage resource allocation** to display the **Manage resource allocation** area.
2. In the **Pool ECPU count** field, select a value that is different than the current value

   By default each instance in an elastic pool is automatically assigned a maintenance window. By selecting a **Pool ECPU count** that is 1024 greater, you have the option of assigning a custom 2-hour maintenance window during which the leader and all elastic pool members are patched together. To select a custom maintenance window for your elastic pool, file a Service Request at Oracle Cloud Support.

See About Scheduled Maintenance and Patching for more information.

3. Click **Apply**.

   When you click **Apply**, the **Lifecycle state** changes to **Scaling in Progress**. After the **Lifecycle state** changes to **Available** the changes apply immediately.

> **Note:**
>
> Decreasing the CPU allocation, **Pool ECPU count**, to a value that cannot accommodate all the members of the elastic pool is not allowed.

For example, for an elastic pool with a **Pool ECPU count** of 256 ECPUs and a pool capacity of 1024 ECPUs: If the elastic pool contains eight (8) Autonomous Database instances with 80 ECPUs each for a total of 640 ECPUs, the elastic pool leader cannot decrease the **Pool ECPU count** to 128 ECPUs. In this case, if the pool size were reduced to 128 ECPUs, the pool capacity would be 512 ECPUs, which is less than the total allocation for the pool members (640 ECPUs).

Create or Join an Elastic Pool While Provisioning or Cloning an Instance

You can create or join an elastic pool when you provision or clone an Autonomous Database instance.

See Provision Autonomous Database for details on how to create an Autonomous Database for your workload type using the Create Autonomous Database dialog.

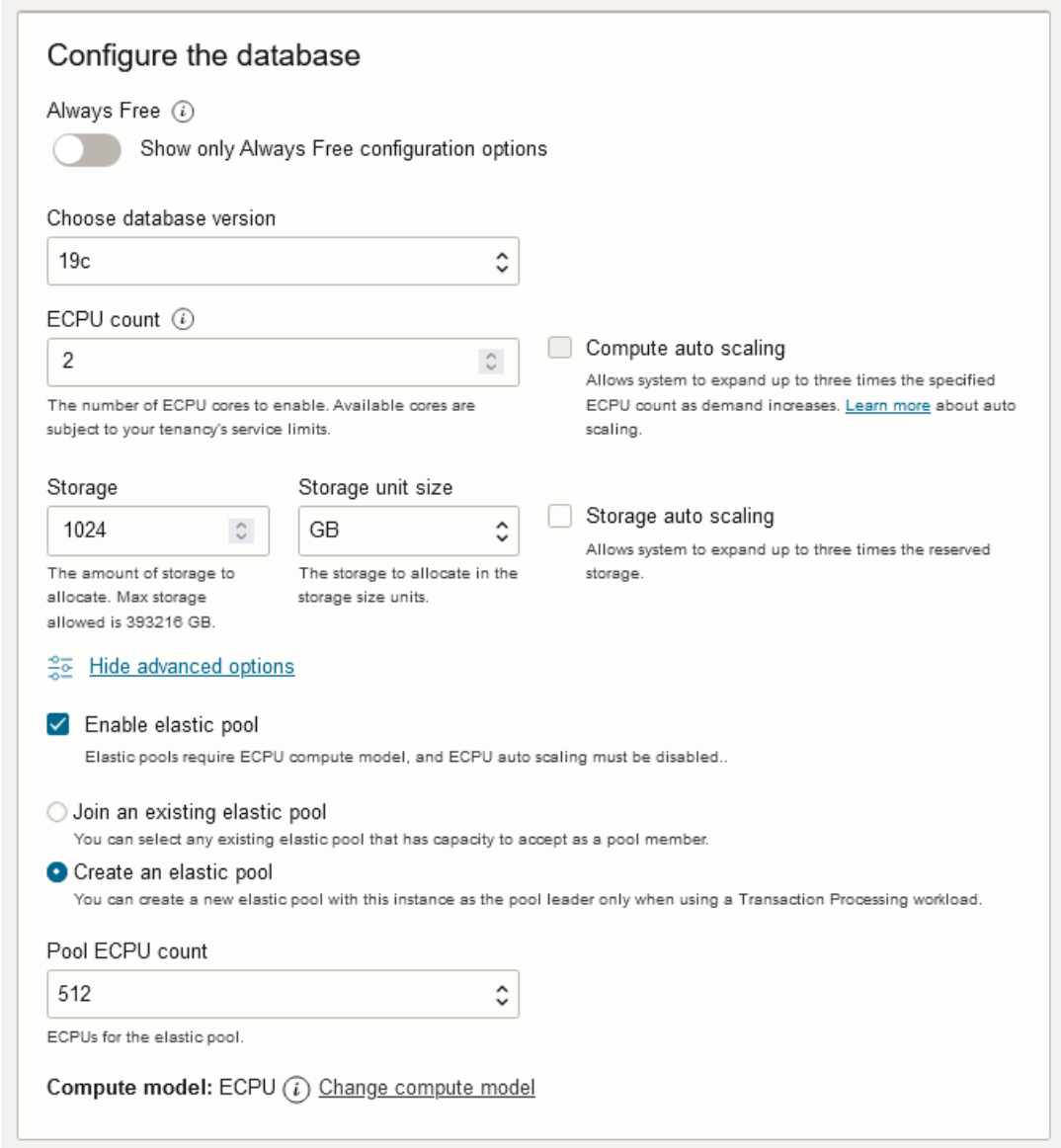
See Clone an Autonomous Database Instance or Clone an Autonomous Database from a Backup for details on cloning.

To create an elastic pool while provisioning or cloning:

> **Note:**
>
> To create an elastic pool the instance must use the ECPU compute model and the workload type you select must be **Transaction Processing**.

1. In the **Configure the database** area, click **Show advanced options** to show advanced options.
2. Deselect **Compute auto scaling**.
3. Select **Enable elastic pool**.
4. Select **Create a elastic pool**.
5. In the **Pool ECPU count** field, select a pool size from the list of pool shapes.

   The valid values that you can select are: 128, 256, 512, 1024, 2048, or 4096.



By default each instance in an elastic pool is automatically assigned a maintenance window. By selecting a **Pool ECPU count** that is 1024 greater, you have the option of assigning a custom 2-hour maintenance window during which the leader and all elastic pool members are patched together. To select a custom maintenance window for your elastic pool, file a Service Request at Oracle Cloud Support.

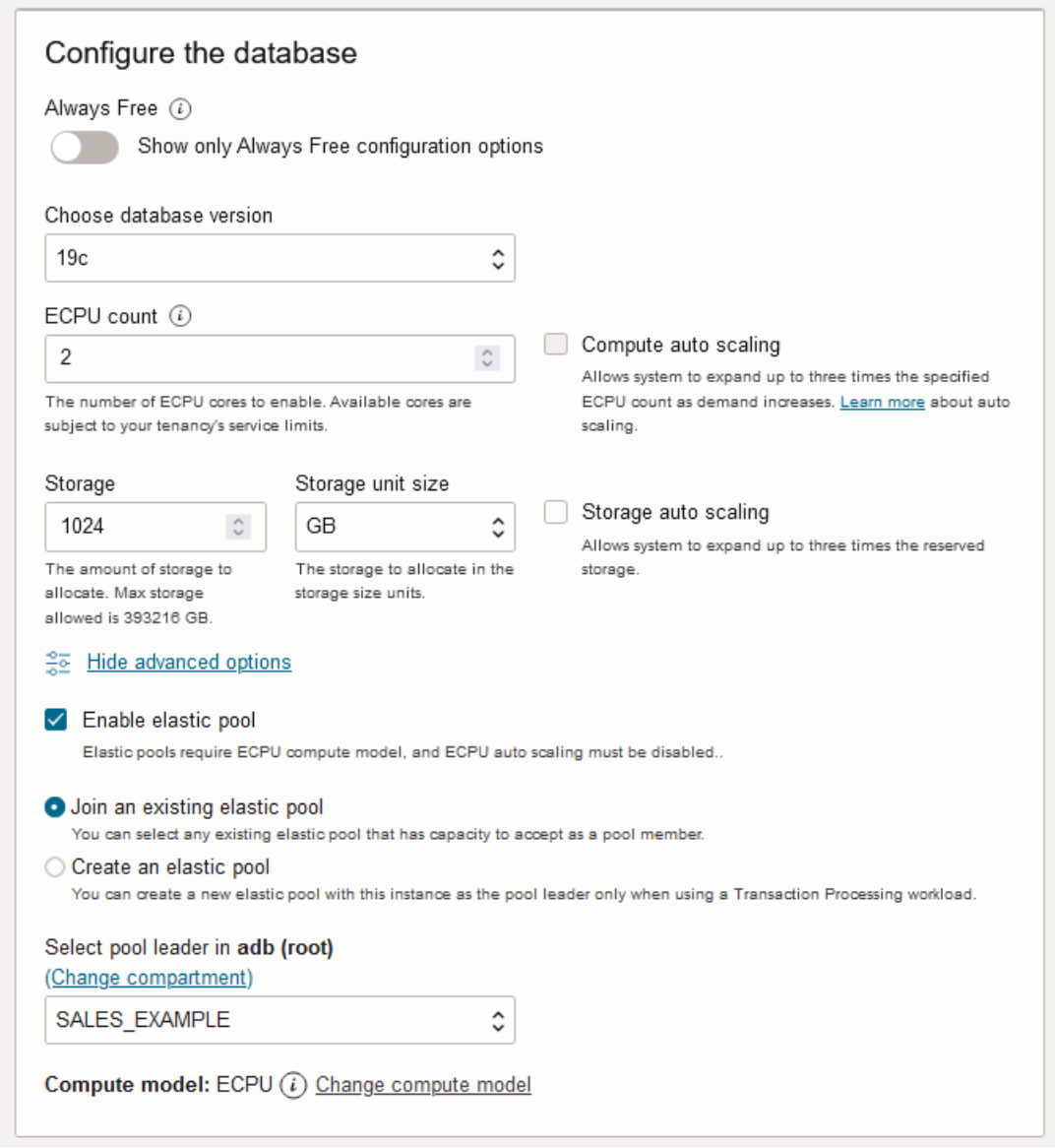
See About Scheduled Maintenance and Patching for more information.

6. Complete the remaining provisioning or cloning steps, as specified in Provision Autonomous Database, Clone an Autonomous Database Instance, or Clone an Autonomous Database from a Backup.

To join an existing elastic pool while provisioning or cloning:

> **Note:**
>
> To join an elastic pool the instance must use the ECPU compute model and the workload type must be one of **Transaction Processing**, **Data Warehouse**, **JSON Database**, or **APEX**.

1. In the **Configure the database** area, click **Show advanced options** to show advanced options.
2. Deselect **Compute auto scaling**.
3. Select **Enable elastic pool**.
4. Select **Join an existing elastic pool**.
5. In the **Select pool leader in *compartment*** field choose a pool leader in a compartment.
   a. Use the compartment shown or click **Change Compartment** to select a different compartment.
   b. Select a pool leader from the list of available pool leaders in the selected compartment.



6. Complete the remaining provisioning or cloning steps, as specified in Provision Autonomous Database, Clone an Autonomous Database Instance, or Clone an Autonomous Database from a Backup.

List Elastic Pool Members

Shows the steps for the pool leader to list elastic pool members.

To list elastic pool members:

1. On the elastic pool leader's Autonomous Database details page, under **Resources**, click **Elastic pool members**.
2. The elastic pool members area shows a list of elastic pool members.

This shows the list of elastic pool members for the leader Autonomous Database instance.

If you click  at the end of a row in the list, you can select an action to perform for the member. The possible actions are:

- **View details**: Shows the member's Oracle Cloud Infrastructure Console
- **Copy OCID**: Copies the member's Autonomous Database instance OCID.
- **Remove from pool**: Brings up a dialog where you can confirm to remove the Autonomous Database instance from the pool.

Remove Pool Members from an Elastic Pool

As an elastic pool member you can remove yourself from an elastic pool. As an elastic pool leader you can remove pool members from an elastic pool.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display name** column.

As a pool member, you can remove your instance from an elastic pool:

1. On the **Details** page, under **Resource allocation**, in the **Elastic pool** field click **Leave pool**.

   This shows the **Leave pool** confirmation dialog.
2. In the **Leave pool** confirmation dialog, enter the database name.
3. Click **Leave**.

   When you click **Leave**, the **Lifecycle state** changes to **Scaling in Progress**. After the **Lifecycle state** changes to **Available** the changes apply immediately.

- As Pool Leader Remove Members from an Elastic Pool
  An elastic pool leader can remove pool members from an elastic pool.
- Notes for Leaving an Elastic Pool
  Provides information about resources when a member or the leader leaves an elastic pool.

As Pool Leader Remove Members from an Elastic Pool

An elastic pool leader can remove pool members from an elastic pool.

1. On the Autonomous Database Details page, under **Resources**, click **Elastic pool members**.

   This shows the **Elastic pool members** area, with a list showing details for each instance that is an elastic pool member.

2. Click  at the end of a row for an instance you want to remove and in the drop down list select **Remove from pool**.

   This shows the **Remove from pool** confirmation dialog.

3. Click **Leave** to confirm.

   When you click **Leave**, the **Lifecycle state** changes to **Scaling in Progress**. After the **Lifecycle state** changes to **Available** the changes apply immediately.

Notes for Leaving an Elastic Pool

Provides information about resources when a member or the leader leaves an elastic pool.

- When a pool member or the leader leaves an elastic pool, auto scaling is disabled. After leaving the elastic pool you can enable auto-scaling for the instance.
- When a pool member leaves an elastic pool, the elastic pool has more resources available. For example, if the elastic pool were fully allocated up to the pool capacity, and an instance with 10 ECPUs leaves the pool, the elastic pool would have 10 available ECPUs.
- Billing for an Autonomous Database instance that leaves an elastic pool returns to individual instance billing, based on the compute and storage resources that the individual instance uses:
  - If a pool member with 2 ECPUs or more leaves the pool, the individual instance's ECPU allocation remains and the instance is billed for that number of ECPUs.
  - If a pool member with 1 ECPU leaves the pool, the ECPU allocation is automatically set to 2 ECPUs and the instance is billed for 2 ECPUs going forward, unless it's scaled up.

Terminate an Elastic Pool

Shows the steps to terminate an elastic pool. Only the pool leader can terminate an elastic pool.

> **Note:**
>
> Terminating an elastic pool is only allowed when there are no pool members in the elastic pool.

- Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then click Autonomous Transaction Processing.

- On the Autonomous Databases page select the Autonomous Database that is the pool leader from the links under the **Display Name** column.

To terminate an elastic pool:

1. On the Autonomous Database Details page, select **Manage resource allocation**.
2. Select **Terminate pool**.
3. Click **Apply** to terminate the elastic pool.

When you click **Apply**, the Lifecycle State changes to **Scaling in Progress**. After the Lifecycle State changes to **Available** the changes apply immediately.

Elastic Pool Notes

Provides information about elastic pool usage.

Notes for elastic pool usage:

- Starting and stopping Autonomous Database instances that are members of an elastic pool does not depend on the state of the leader. That is, you can independently stop and start each instance that is part of an elastic pool, including the leader and members of the elastic pool that are not the leader.
- By default each instance in an elastic pool is automatically assigned a maintenance window. By selecting a pool shape that is 1024 ECPUs or greater, you have the option of assigning a custom 2-hour maintenance window during which the leader and all elastic pool members are patched together. To select a custom maintenance window for your elastic pool, file a Service Request at Oracle Cloud Support.

Migrate Oracle Databases to Autonomous Database

Oracle provides options to migrate your database to Autonomous Database.
You can choose one of these options depending on the amount of data you want to migrate, downtime requirements during the migration, network bandwidth between your source database and your Oracle Cloud Infrastructure region, and whether you want manual control over the migration.

- Migration Prerequisites
  Before starting your migration, it is recommended to run the Cloud Premigration Advisor Tool (CPAT), which helps you evaluate your source database for compatibility with the Autonomous Database.
- Recommended Migration Methods
  Following are the recommended migration methods to migrate your database into an Autonomous Database.

Migration Prerequisites

Before starting your migration, it is recommended to run the Cloud Premigration Advisor Tool (CPAT), which helps you evaluate your source database for compatibility with the Autonomous Database.

CPAT identifies potential actions you may need to take before or during the migration, prioritizing their importance and suggesting resolutions. Some migration tools and services automatically run this advisor.

See Cloud Premigration Advisor Tool for more information.

Recommended Migration Methods

Following are the recommended migration methods to migrate your database into an Autonomous Database.

Online Migration

If your source database needs to be always online and you can only take minimal downtime for the migration, you can use one of the online migration services and tools below, independent of your database size. Note that both options require the source database to be on version 11.2.0.4 or later. If your source database is using an older version, you need to upgrade it first if you want to use one of the online migration methods.

Following are the online migration methods:

- **OCI Database Migration**: OCI Database Migration, a managed cloud service, provides validated, cross-version, fault-tolerant, and incremental Oracle Database migrations for online and offline use cases. You can choose this option if you want a fully managed cloud service that handles the migration for you.

  See OCI Database Migration for more information.

- **Zero Downtime Migration (ZDM)**: Zero Downtime Migration (ZDM) is a tool with a command line interface that you install and run on a host you provision. Zero Downtime Migration provides online and offline migration options. You can use this option if you want more control over the migration.

  See Zero Downtime Migration for more information.

> **Note:**
>
> These methods also run the Cloud Premigration Advisor Tool (CPAT) to check your source database for compatibility with the Autonomous Database and use Oracle Data Pump and Oracle GoldenGate to carry out the migration for you.

Offline Migration

If you can take your source database offline for the migration, you can use one of the following migration options in addition to OCI Database Migration and Zero Downtime Migration (ZDM).

> **Note:**
>
> This is a manual method; you must orchestrate the export/import manually.

Following is the offline migration method:

- **Oracle Data Pump**: Oracle Data Pump technology enables very high-speed movement of data and metadata from one database to another. You can export your source database or individual schemas as dump files and import them into the Autonomous Database. You can store the dump files on the Object Store or a network filesystem (NFS) that you can attach to your source database and to your Autonomous Database.

  See Overview of Oracle Data Pump for the product documentation for Oracle Data Pump.

See Import Data Using Oracle Data Pump on Autonomous Database for more information about how to export and import your source schemas into the Autonomous Database.

Other Migration Methods

In addition to the recommended migration tools and services, you can use the following offline data migration methods:

> **Note:**
>
> These methods require more manual steps, coding, and advanced knowledge of each method. You must also migrate your source schema metadata using your own `DDL` scripts or tools such as Oracle Data Pump before initiating the data load using these methods. These options may be preferable for small data sets for which you want to avoid installing additional tools or using cloud migration services.

Following are the other offline migration methods:

- **Database Links**: You can create database links from your Autonomous Database to your source database and insert data into your tables by running queries over the database link. You can use multiple sessions to load multiple tables concurrently to improve the performance of the data load.

  See Create Database Links from Autonomous Database to Another Autonomous Database for more information about creating database links from your Autonomous Database to your source databases.

- **Unload and Load Using Files**: You can unload data from your source database into files and load those files into your target tables in your Autonomous Database. Autonomous Database supports loading data from different file formats including CSV, JSON, Parquet, and more. You can place the files on the Object Store or a network filesystem (NFS) attached to your Autonomous Database.

  See Load Data from Files in the Cloud for more information about how to load data from files in the object store.

  See Attach Network File System to Autonomous Database for attaching NFS directories to your Autonomous Database.

  See Load Data or Query Data from Files in a Directory for more information about loading data from files on a filesystem.

- **Materialized Views**: Using database links from your Autonomous Database to your source database, you can also create materialized views in your target database. As changes happen on your source tables, you can refresh the materialized views to keep them updated during the migration.

  See Managing Read-Only Materialized Views for more information about using materialized views to replicate data.

> **Note:**
>
> Deciding the migration tool or service depends on multiple factors such as your source database, source data format, data volume, and complexity. To help you identify the most optimal solution for migrating your data to the Autonomous Database, Oracle provides an advisory utility called Oracle Cloud Migration Advisor. See Migrate Oracle Databases to OCI for more information about this utility.

6
Reference

- Previous Feature Announcements
  Announcements for the important changes made to Oracle Autonomous Database Serverless.
- Sample Star Schema Benchmark (SSB) Queries and Analytic Views
  The SSB schema contains the tables: lineorder, customer, supplier, part, and dwdate. The following is a list of sample queries and analytic views you can use against the SSB schema. Note that you need to prefix the table names with the schema name SSB in your queries.
- Autonomous Database Supplied Package Reference
  This appendix provides information about the packages you use with Autonomous Database. The `DBMS_CLOUD` topic also covers the `DBMS_CLOUD REST APIs`.
- Autonomous Database for Experienced Oracle Database Users
  This appendix provides information on using Autonomous Database for experienced Oracle Database users with Autonomous Database Serverless.
- Migrating MySQL and Third-Party Databases to Autonomous Database
- SODA Collection Metadata
  It describes SODA Collection metadata on the database.
- Cloud Support and Identity Information
  This lists the procedure to file a service request using Oracle Cloud Support.

Previous Feature Announcements

Announcements for the important changes made to Oracle Autonomous Database Serverless.

See What's New for Oracle Autonomous Database Serverless for feature announcements in 2023.

- 2023 What's New
  Announcements for the noteworthy changes made to Oracle Autonomous Database Serverless in the year 2023.
- 2022 What's New
  Announcements for the noteworthy changes made to Oracle Autonomous Database Serverless in the year 2022.
- 2021 What's New
  Announcements for the noteworthy changes made to Oracle Autonomous Database Serverless Deployment in the year 2021.
- 2020 What's New
  Announcements for the noteworthy changes made to Oracle Autonomous Database Serverless Deployment in the year 2020.
- 2019 What's New
  Announcements for the noteworthy changes made to Oracle Autonomous Database Serverless Deployment in the year 2019.

- 2018 What's New
  Announcements for the noteworthy changes made to Oracle Autonomous Database Serverless Deployment in the year 2018.

2023 What's New

Announcements for the noteworthy changes made to Oracle Autonomous Database Serverless in the year 2023.

- December 2023
- November 2023
- October 2023
- September 2023
- August 2023
- July 2023
- June 2023
- May 2023
- April 2023
- March 2023
- January 2023
- February 2023

December 2023

Feature	Description
Cross Tenancy Cloning	You can clone an Autonomous Database instance from one tenancy, source tenancy, to a different tenancy (destination tenancy). The cross tenancy cloning option is only available using the CLI or the Autonomous Database REST APIs. This option is not available using the Oracle Cloud Infrastructure Console. See Cross Tenancy and Cross-Region Cloning for more information.
Invoke Azure Remote Functions	You can invoke Azure remote functions in your Autonomous Database as SQL functions. See Steps to Invoke Azure Function as SQL Functions for more information.
Create Database Links to Oracle RAC Database	You can create database links from an Autonomous Database to a target Oracle RAC database and specify multiple hostnames. See Create Database Links from Autonomous Database to an Oracle Database on a Private Endpoint for more information.
New Cloud Links Views	Obtain information on the Cloud Link specific privileges granted to all users or to the current user. See Monitor and View Cloud Links Information for more information.
Documentation Addition: Load Data from AWS S3	Provides an example for loading data into an Autonomous Database instance from AWS S3. See Load Data into Autonomous Database from AWS S3 for more information.

November 2023

Feature	Description
Oracle APEX 23.2	Autonomous Database uses Oracle APEX Release 23.2. See Create Applications with Oracle APEX in Autonomous Database for more information.
Select AI with Azure OpenAI Service	Autonomous Database can interact with AI service providers. Select AI now supports the Azure OpenAI Service, OpenAI, and CohereAI. One way LLMs can work with Oracle database is by generating SQL from natural language prompts that allow you to talk to your database. See Use Select AI to Generate SQL from Natural Language Prompts for more information.
Break Glass Access with `SAAS_ADMIN` User	Autonomous Database supports break glass access for SaaS providers. Break glass access allows a SaaS operations team, when explicitly authorized by a SaaS customer, to access a customer's database to perform critical or emergency operations. See Break Glass Access for SaaS on Autonomous Database for more information.
Database Links Without a Wallet (TLS)	You can create database links from one Autonomous Database instance to a publicly accessible Autonomous Database without a wallet (TLS). See Create Database Links from Autonomous Database to a Publicly Accessible Autonomous Database without a Wallet (TLS) for more information.

October 2023

Feature	Description
Vault Secret Credential Objects	You can use vault secret credentials to access cloud resources or to access other databases (use vault secret credentials anywhere where username/password type credentials are required). Supported vaults are: • Oracle Cloud Infrastructure Vault • Azure Key Vault • AWS Secrets Manager • GCP Secret Manager See Use Vault Secret Credentials for more information.
Documentation Addition: Billing Information	Documentation includes summary and detailed Autonomous Database billing information. See How Is Autonomous Database Billed? for more information.
Documentation Addition: Oracle Database Migration Information	Documentation includes information on migrating your Oracle Database to Autonomous Database. See Migrate Oracle Databases to Autonomous Database for more information.
ServiceNow Support for Database Links with Oracle-Managed Heterogeneous Connectivity	Autonomous Database support for Oracle-managed heterogeneous connectivity makes it easy to create database links to ServiceNow. When you use database links with Oracle-managed heterogeneous connectivity, Autonomous Database configures and sets up the connection. See Create Database Links to Non-Oracle Databases with Oracle-Managed Heterogeneous Connectivity for more information.

Feature	Description
Invoke Oracle Cloud Infrastructure Functions or AWS Lambda Remote Functions	You can invoke Oracle Cloud Infrastructure Functions and AWS Lambda remote functions in your Autonomous Database as SQL functions. See Invoke User Defined Functions for more information.

September 2023

Feature	Description
Use Select AI to Generate SQL from Natural Language Prompts	Autonomous Database can interact with AI service providers including: OpenAI and CohereAI to provide access to Generative AI capability from Oracle Database that uses Large Language Models (LLMs). One way LLMs can work with Oracle database is by generating SQL from natural language prompts that allow you to talk to your database. See Use Select AI to Generate SQL from Natural Language Prompts for more information.
Cloud Links Usability and Security	Cloud Links provide a cloud-based method to collaborate between Autonomous Databases. Cloud Links now allow you to set a data set owner. Cloud Links can be configured to require an additional authorization step, where a data set only allows specified databases to access the data set. When you register a data set, you can also specify refreshable clones to offload access to the data set. See Use Cloud Links for Read Only Data Access on Autonomous Database for more information.
Elastic Pools	Use an elastic pool to consolidate your Autonomous Database instances, in terms of their allocation of compute resources, and to provide up to 87% cost savings. See Use and Manage Elastic Pools on Autonomous Database for more information.
Autonomous Database Free Container Image	Use the Oracle Autonomous Database Free Container Image to run Autonomous Database in a container in your own environment, without requiring access to the Oracle Cloud Infrastructure Console or to the internet. See Use the Oracle Autonomous Database Free Container Image for more information.
ECPU-based Databases with Transaction Processing Workload Type Support per-Gigabyte Increments of Storage	You may now provision, clone, and scale using Gigabyte (GB) increments of storage for ECPU-based Autonomous Databases provisioned with the Transaction Processing workload type. The minimum storage allowed for an Autonomous Database instance in ECPU compute model with Transaction Processing workload is 20 GB. See Provision Autonomous Database for more information.
Google Analytics Support for Database Links with Oracle-Managed Heterogeneous Connectivity	Use Autonomous Database Oracle-managed heterogeneous connectivity to create database links to Google Analytics. When you use database links with Oracle-managed heterogeneous connectivity, Autonomous Database configures and sets up the connection. See Create Database Links to Non-Oracle Databases with Oracle-Managed Heterogeneous Connectivity for more information.

August 2023

Feature	Description
DBMS_SHARE Package	You can use the subprograms and views in the DBMS_SHARE package to share data with external systems. See DBMS_SHARE Package for more information.
Logical Partition Change Tracking (LPCT)	Logical Partition Change Tracking enables you to create logical partitions on base tables. It evaluates the staleness of the base tables for individual logical partitions without using a materialized view log or requiring any of the tables used in the materialized view to be partitioned. Logical Partition Change Tracking provides the capability to leverage the user-supplied logical partitioning information of base tables of materialized view for a more fine-grained, partition-level tracking of stale data for both refresh and rewrite purposes. While classical Partitioning Change Tracking (PCT) relies on the physical partitioning of tables, LPCT has no dependency on tables being physically partitioned; you can use LPCT with both partitioned and nonpartitioned tables. See Logical Partition Change Tracking and Materialized Views for more information.
Update OCPU to ECPU Compute Model	You can update an Autonomous Database instance from the OCPU billing model to the ECPU billing model. See Update to ECPU Billing Model on Autonomous Database for more information.
Choose Backup Retention Period with ECPU Compute Model	With the ECPU billing model you have the option to select the backup retention period for automatic backups, with a retention period between 1 day and up to 60 days. See Edit Automatic Backup Retention Period on Autonomous Database for more information.
AWS Glue Data Catalog Integration	Autonomous Database allows you to synchronize with Amazon Web Services (AWS) Glue Data Catalog metadata. You can query data stored in S3 from Autonomous Database without having to manually derive the schema for the external data sources and create external tables. See Query External Data with AWS Glue Data Catalog for more information.

July 2023

Feature	Description
Track Table Changes with Flashback Time Travel	Use Flashback Time Travel to view past states of database objects or to return database objects to a previous state without using point-in-time media recovery. See Tracking Table Changes with Flashback Time Travel for more information.
Database Actions Quick Links	Database Actions on the Oracle Cloud Infrastructure Console provides quick links to select Database Actions cards or to select the Database Actions Launchpad. See Access Database Actions as ADMIN for more information.

Feature	Description
Object Storage and REST Endpoint Management	DBMS_CLOUD supports a number of preconfigured Object Store and REST endpoints. DBMS_CLOUD also allows you to access additional customer-managed endpoints. See DBMS_CLOUD Endpoint Management for more information.
detectfieldorder DBMS_CLOUD Format Option	The detectfieldorder format option specifies that the fields in external data files are in a different order than the columns in the table. Specify this option with DBMS_CLOUD.COPY_DATA, DBMS_CLOUD.CREATE_EXTERNAL_TABLE, DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE, or DBMS_CLOUD.CREATE_HYBRID_PART_TABLE to detect the order of fields using the first row of each external data file and map it to the columns of the table. See DBMS_CLOUD Package Format Options for more information.
Vault Secret Credential Objects	You can create vault secret credentials with secrets stored in Oracle Cloud Infrastructure Vault. You can use vault secret credentials to access cloud resources or to access other databases (use vault secret credentials anywhere where username/password type credentials are required). See Use Vault Secret Credentials for more information.
Export to Directory with DBMS_CLOUD.EXPORT_DATA	Export files as text or as Oracle Data Pump dump files to a directory. See Export Data to a Directory for more information.

June 2023

Feature	Description
Oracle Cloud Infrastructure Signing Key Based Authentication for Oracle Data Pump Import	Oracle Data Pump import using impdp supports credentials created using Oracle Cloud Infrastructure key based attributes. See Import Data Using Oracle Data Pump on Autonomous Database for more information.
Configurable Automatic Failover Data Loss Limit for a Local Standby Database	For Autonomous Data Guard you can specify an automatic failover data loss limit between 0 and 3600 seconds. Autonomous Data Guard performs automatic failover to a local standby database when a local standby database is available and the system can guarantee either the default zero data loss RPO or up to the data loss limit you specify. See Automatic Failover with a Standby Database for more information.
Oracle Workspace Manager	Oracle Workspace Manager provides an infrastructure that enables applications to create workspaces and group different versions of table row values in different workspaces. Use Oracle Workspace Manager to version-enable one or more user tables in the database. See Using Oracle Workspace Manager on Autonomous Database for more information.
Automatic Failover Begin and Automatic Failover End Events	The AutonomousDatabase-AutomaticFailoverBegin and AutonomousDatabase-AutomaticFailoverEnd events are generated when an automatic failover begins and ends. These events are only be triggered if you are using Autonomous Data Guard. See About Critical Events on Autonomous Database for more information.

Feature	Description
Operator Access Event	The `OperatorAccess` event is generated when operator access is detected for the database. Autonomous Database operations team never access your data unless you explicitly grant permission through a Service Request for a specified duration. See Information Events on Autonomous Database for more information.
Built-in Tool Availability Service Level Objective (SLO)	Oracle will use commercially reasonable efforts to have the listed built-in tools meet the Availability objectives as defined in Service Level Objective (SLO) documentation. See Availability Service Level Objectives (SLOs) for more information.
CMU-AD with Active Directory Servers on a Private Endpoint	There are now two options for configuring Autonomous Database with Centrally Managed Users (CMU) with Microsoft Active Directory: • Active Directory (AD) servers publicly accessible: the Active Directory servers are accessible from Autonomous Database through the public internet. • Active Directory (AD) servers reside on a private endpoint: the Active Directory servers reside on a private endpoint and are not accessible from Autonomous Database through the public internet. See Use Microsoft Active Directory with Autonomous Database for more information.

May 2023

Feature	Description
Oracle APEX 23.1	Autonomous Database uses Oracle APEX Release 23.1. See Creating Applications with Oracle APEX in Autonomous Database for more information.
Google BigQuery Support for Database Links with Oracle-Managed Heterogeneous Connectivity	Autonomous Database support for Oracle-managed heterogeneous connectivity to create database links to Google BigQuery. When you use database links with Oracle-managed heterogeneous connectivity, Autonomous Database configures and sets up the connection. See Create Database Links to Non-Oracle Databases with Oracle-Managed Heterogeneous Connectivity for more information.
Log File Options for `DBMS_CLOUD`	`DBMS_CLOUD` provides `format` parameter options for logging customization: `enablelogs`, `logprefix`, `logdir`, and `logretention`. See DBMS_CLOUD Package Format Options for more information.
`RESULT_CACHE_MODE` Parameter is Modifiable at Session and System Level	Set the `RESULT_CACHE_MODE` parameter to specify which queries are eligible to store result sets in the result cache. Only query execution plans with the result cache operator will attempt to read from or write to the result cache. See RESULT_CACHE_MODE for more information.
Stripe Financial Views	Stripe is an online payment processing and credit card processing platform for businesses. The Stripe views allow you to query views created on top of Stripe APIs with `DBMS_CLOUD` to get Stripe information such as products, invoices, plans, accounts, subscriptions, and customers. See Stripe Views on Autonomous Database for more information.

Feature	Description
Oracle GoldenGate Parallel Replicat (Integrated Mode)	Integrated Parallel Replicat (iPR) supports GoldenGate replication apply for procedural replication, Automatic CDR, and DML handlers. See Deciding Which Apply Method to Use and About Parallel Replicat for more information.
Persistent Messaging in `DBMS_PIPE`	The `DBMS_PIPE` package has extended functionality on Autonomous Database to support persistent messaging, where messages are stored in Cloud Object Store. See Use Persistent Messaging with Messages Stored in Cloud Object Store for more information.
Access Oracle Cloud Infrastructure Logging Data with Autonomous Database Views	Using the Oracle Cloud Infrastructure Logging Interface you can access log data from an Autonomous Database instance, in relational format. You can query log data in Oracle Cloud Infrastructure across all compartments and regions. See Oracle Cloud Infrastructure Logging Interface Views for more information.

April 2023

Feature	Description
Oracle Spatial Features for Geocoding Address Data and Reverse Geocoding Location Data	Oracle Spatial on Autonomous Database includes features for geocoding address data and for reverse geocoding longitude/latitude data to a street address. See Using Oracle Spatial with Autonomous Database for more information.
Kerberos Authentication for CMU-AD	You can configure Autonomous Database to use Kerberos authentication for CMU with Microsoft Active Directory users. This configuration allows CMU Active Directory (CMU-AD) users to access an Autonomous Database instance using Kerberos credentials.
User Defined Notification Handler for Scheduler Jobs	Database Scheduler provides an email notification mechanism to track the status of periodically running or automated jobs. In addition to this, the Database Scheduler also supports user-defined PL/SQL Scheduler job notification handler procedure. Adding a scheduler job notification handler procedure allows you to monitor scheduled or automated jobs running in your Autonomous Database. See User Defined Notification Handler for Scheduler Jobs for more information.
Database Links with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint	You can create database links from an Autonomous Database to an customer-managed Oracle Database Gateway to access Non-Oracle databases that are on a private endpoint. See Create Database Links with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint for more information.
Oracle Machine Learning Notebooks Early Adopter	Oracle Machine Learning Notebooks Early Adopter is an enhanced web-based notebook platform for data engineers, data analyst, R and Python users, and data scientists. You can write code, text, create visualizations, and perform data analytics including machine learning. Notebooks work with interpreters in the back-end. In Oracle Machine Learning, notebooks are available in a project within a workspace, where you can create, edit, delete, copy, move, and even save notebooks as templates. See Get Started with Notebooks Early Adopter for Data Analysis and Data Visualization for more information.

Feature	Description
Salesforce Support for Database Links with Oracle-Managed Heterogeneous Connectivity	Autonomous Database support for Oracle-managed heterogeneous connectivity to create database links to Salesforce databases. When you use database links with Oracle-managed heterogeneous connectivity, Autonomous Database configures and sets up the connection. See Create Database Links to Non-Oracle Databases with Oracle-Managed Heterogeneous Connectivity for more information.
Snapshot Standbys for Cross-Region Disaster Recovery	A disaster recovery cross-region peer can be converted to a snapshot standby. This converts either an Autonomous Data Guard cross-region standby database or a cross-region Backup-Based Disaster Recovery peer to a read-write database for up to two days. See Convert Cross Region Peer to Snapshot Standby for more information.
Singleton Pipe in `DBMS_PIPE`	Singleton Pipe is an addition to the `DBMS_PIPE` package that allows you to cache a custom message. Using a Singleton Pipe you can send and retrieve a custom message and share the message across multiple database sessions with concurrent reads. See Caching Messages with Singleton Pipes for more information.
Migrate Databases Across Regions with Minimal Downtime Using Autonomous Data Guard	You may now terminate, as well as perform all other Primary database actions on a remote Autonomous Data Guard standby when it becomes the Primary database, after a switchover or after a failover. This enables you to use a remote peer for migrating your database to the remote region. See Using Standby Databases with Autonomous Data Guard for Disaster Recovery for more information.
Send Microsoft Teams Notifications	Use Microsoft Teams notifications. You can send messages, alerts, or the output of a query from Autonomous Database to a Microsoft Teams Channel. See Send Microsoft Teams Notifications from Autonomous Database for more information.
Send Email Using `DBMS_CLOUD_NOTIFICATION`	You can send email to a public SMTP endpoint using the Oracle Cloud Infrastructure Email Delivery service. See Send Email from Autonomous Database Using DBMS_CLOUD_NOTIFICATION for more information.

March 2023

Feature	Description
Cloud Links	Cloud Links provide a cloud-based method to collaborate between Autonomous Databases. Using Cloud Links you can access read-only data based on the cloud identities of Autonomous Database instances. The scope of collaboration can be the region where an Autonomous Database resides, individual tenancies, compartments, or particular Autonomous Database instances. See Using Cloud Links for Read Only Data Access on Autonomous Database for more information.
Backup-Based Disaster Recovery	Backup-Based Disaster Recovery uses backups to instantiate a peer database at the time of switchover or failover. For local Backup-Based Disaster Recovery, existing local backups are utilized and there are no additional costs. Cross-Region Backup-Based Disaster Recovery is also available and incurs additional costs. See Using Backup-Based Disaster Recovery for more information.

Feature	Description
Documentation Addition: Oracle Machine Learning with Autonomous Database	Provides information on options for using Oracle Machine Learning with Autonomous Database. See Machine Learning with Autonomous Database for more information.
Export Data to Cloud Object Storage as Parquet	Use `DBMS_CLOUD.EXPORT_DATA` to export data as text. The text format export options are CSV, JSON, Parquet, or XML. See Export Data to Object Store as Text (CSV, JSON, Parquet, or XML) for more information.
Oracle Real Application Testing Capture and Replay	Use Capture and Replay from one Autonomous Database instance to another Autonomous Database instance. This enables you to compare workloads across different Autonomous Database instances. These Autonomous Database instances may vary at patch levels, database versions, or regions. See Using Oracle Real Application Testing for more information.
Availability Domain Information	You can obtain tenancy details, including the Availability Domain (AD) for an Autonomous Database instance. See Obtain Tenancy Details for more information.
Provision an Autonomous Database Instance and Save as Stack	The provisioning option Save as Stack allows you to create and deploy an Autonomous Database instance using Resource Manager. See Provision Autonomous Database and Overview of Resource Manager for more information.
Oracle Client for Microsoft Tools (OCMT)	Oracle Client for Microsoft Tools (OCMT) is a graphical user interface (GUI) native Microsoft Software installer (MSI) that simplifies ODP.NET setup and provides Autonomous Database connectivity to multiple Microsoft data tools. See Connect Power BI and Microsoft Data Tools to Autonomous Database for more information.
Long-Term Backups	You can create long-term backups on Autonomous Database with a retention period between three months and up to ten years. When you create a long-term backup you can create a one-time backup or set a schedule to automatically create backups that are taken weekly, monthly, quarterly, or annually (yearly). See About Backup and Recovery on Autonomous Database and Create Long-Term Backups on Autonomous Database for more information.
Use Directories to Load Data with `DBMS_CLOUD` Procedures	As an alternative to an object store location URI, you can specify a directory with `DBMS_CLOUD` procedures to load or unload data from files in a directory, including directories created on attached network file systems. See Load Data from Directories in Autonomous Database for more information.

January 2023

Feature	Description
ECPU Compute Model	Autonomous Database offers two compute models when you create or clone an instance: ECPU and OCPU. See Compute Models in Autonomous Database for more information.

Feature	Description
Configure Built-in Tools	Autonomous Database includes built-in tools that you can enable and disable when you provision or clone a database, or at any time for an existing database. See Managing Autonomous Database Built-in Tools for more information.
Configure a Custom Private IP Address for a Private Endpoint	You can configure a private endpoint when you provision or clone an Autonomous Database instance, or if your instance is configured to use a public endpoint you can change the configuration to use a private endpoint. Optionally when you configure a private endpoint you can enter a custom private IP address. Providing a custom IP address allows you to preserve an existing private IP address when you migrate from another system. See Configure Private Endpoints for more information.
Cross-Region Clone from Backup	Autonomous Database provides the option to create a clone in another region when you are cloning a database and you select **Clone from a backup**. In this case, you choose your preferred region to select the region where you want to create the clone. See Clone an Autonomous Database from a Backup for more information.
Send Slack Notifications	You can send slack notifications from Autonomous Database. This allows you to send messages to a Slack channel or send the results of a query to a Slack channel. See Send Slack Notifications from Autonomous Database for more information.

February 2023

Feature	Description
Maximum Storage Space Metric	The `StorageMax` metric shows the maximum amount of storage reserved for the database. See Autonomous Database Metrics and Dimensions for more information.
Oracle Cloud Infrastructure Vault Secret for ADMIN Password	When you create or clone an Autonomous Database instance or when you reset the ADMIN password, you can use an Oracle Cloud Infrastructure vault secret to specify the ADMIN password. See Use Oracle Cloud Infrastructure Vault Secret for ADMIN Password for more information.
Inactive Connections Detected Event	The `InactiveConnectionsDetected` event is generated when the number of inactive database connections detected is above a certain ratio, compared to all database connections for the Autonomous Database instance. Subscribing to this event can help you keep track of unused connections. See About Information Events on Autonomous Database for more information.
Enable Automatic Time Zone File Updates	Autonomous Database lets you choose to enable the feature that automatically updates time zone files on your Autonomous Database instance. See Manage Time Zone File Version on Autonomous Database for more information.

Feature	Description
Database Relocate and Failed Logon Events	The `FailedLoginWarning` event is generated when the number of failed login attempts reaches 3 * number of total users in the last three (3) hours. The `InstanceRelocateBegin` and `InstanceRelocateEnd` events are triggered when an Autonomous Database instance is relocated to another Exadata infrastructure due to server maintenance, hardware refresh, a hardware issue, or as part of a resource scale up. See About Critical Events on Autonomous Database for more information.

2022 What's New

Announcements for the noteworthy changes made to Oracle Autonomous Database Serverless in the year 2022.

December 2022

Feature	Description
Data Pipelines for Loading and Exporting	Data pipelines allow you to repeatedly load data from object store, or export data to object store. Load pipelines provide continuous incremental data loading from external sources (as data arrives on object store it is loaded to a database table). Export pipelines provide continuous incremental data exporting to object store (as new data appears in a database table it is exported to object store). See Using Data Pipelines for Continuous Load and Export for more information.
Credential Objects for use with `UTL_HTTP`, `UTL_SMTP`, and `DBMS_LDAP`	You can use credential objects to set authentication for use with `UTL_HTTP`, `UTL_SMTP` or `DBMS_LDAP`. See Use Credential Objects to set SMTP Authentication, Use Credential Objects to Set HTTP Authentication, and PL/SQL Packages for more information.

Feature	Description
Medium and High Service Limits with Auto Scaling	The number of concurrent statements for the high and medium services increases by three times when OCPU Auto Scaling is enabled. See Service Concurrency for more information.
Multiple Data Catalog Support	Instead of each Autonomous Database instance supporting a single connection to a Data Catalog instance, this feature allows connections to multiple Data Catalog instances. See About Querying with Data Catalog for more information.

November 2022

Feature	Description
Cloud Code Repository Branch Management and Schema Export and Install	Additions and improvements for Cloud Code Repository in the `DBMS_CLOUD_REPO` package for (Git) repository branch management and schema export and install. See Using and Managing a Cloud Code Repository with Autonomous Database for more information.
Log File Prefix and Retention Format Options for `COPY_DATA`	`DBMS_CLOUD.COPY_DATA` supports the format parameter options: `logretention` and `logprefix`. See DBMS_CLOUD Package Format Options for more information.
Cross-Region Autonomous Data Guard: Remove Second Hostname from Database Connection Strings	The `wallet.zip` and the connection strings provided for a database with cross-region Autonomous Data Guard enabled no longer contain both the primary and standby database hostnames in a single connection string or wallet. To avoid connection retry delays and optimize your disaster recovery setup, it is recommended to use the Primary database wallet or connection string to connect to the Primary database, and the remote database wallet or connection string to connect to the remote database (when the remote database is Available to connect to, after a switchover or failover). See Cross-Region Autonomous Data Guard Connection Strings and Wallets for more information.
Data Transforms in Database Actions	Data Transforms is a built-in data integration tool and is accessible from Database Actions. Data Transforms is a simple-to-use, drag-and-drop, no-code tool to load and transform data from heterogeneous sources to Autonomous Database. Use Data Transforms for all data integration needs such as building data warehouses and feeding analytics applications. See The Data Transforms Page for more information.
Access Network File System (NFS) Directories	You can attach a Network File System to a directory location in your Autonomous Database. This allows you to load data from Oracle Cloud Infrastructure File Storage in your Virtual Cloud Network (VCN) or from any other Network File System in on-premises data centers. See Access Network File System from Autonomous Database for more information.
Customer Managed Keys with Cross-Region Autonomous Data Guard	Autonomous Database fully supports using customer-managed keys with a remote standby database with Autonomous Data Guard. See Autonomous Data Guard with Customer Managed Keys and Notes for Customer-Managed Keys with Autonomous Data Guard for more information.

Feature	Description
Cross-Region Refreshable Clones	Create one or more clones in regions other than the region of your primary (source) database. Clones in remote regions can be refreshed from the source database. See Using Refreshable Clones with Autonomous Database for more information.
Use Text Index to Query Text on Object Storage	You can build an Oracle Text index on object store files. This allows you to search the object store for text and use wildcards with your search. See Query Text in Object Storage for more information.
Documentation Addition: Data Lake Capabilities with Autonomous Database	A new section provides information on using Autonomous Database as a data lakehouse. See Using Data Lake Capabilities with Autonomous Database for more information.
Bulk Operations for Files in the Cloud	The PL/SQL package `DBMS_CLOUD` offers parallel execution support for bulk file upload, download, copy, and transfer activities, which streamlines the user experience and delivers optimal performance for bulk file operations. See Bulk Operations for Files in the Cloud for more information.

October 2022

Feature	Description
Clone from Backup Select Latest Available Timestamp Option	When you clone an Autonomous Database instance you can select clone from latest backup. This option selects the latest available backup timestamp as the clone source. You can select this option if a database becomes unavailable and you want to create a clone, or for any reason when you want to create a clone based on the most recent backup. See Clone Autonomous Database from a Backup for more information.
Use Customer-Managed Encryption Keys Located in a Different Tenancy	You can use customer-managed master encryption keys with the Vault and Keys in a different tenancy. The Vault and the Autonomous Database instance can be in a different tenancy, but they must be in the same region. See Use Customer-Managed Encryption Key Located in a Remote Tenancy for more information.
Use Google Service Account to Access GCP Resources	You can use a Google service account to access Google Cloud Platform (GCP) resources from an Autonomous Database instance. See Use Google Service Account to Access Google Cloud Platform Resources for more information.
`SYSDATE_AT_DBTIMEZONE` Parameter is Modifiable at Session and System Level	Depending on the value of `SYSDATE_AT_DBTIMEZONE`, you see either the date and time based on the default Autonomous Database time zone, Coordinated Universal Time (UTC), or based on the time zone that you set in your database. See SYSDATE_AT_DBTIMEZONE for more information.
View Oracle Cloud Operations Actions	The `DBA_OPERATOR_ACCESS` view stores information on the actions that Oracle Cloud Infrastructure cloud operations performs on your Autonomous Database. This view will not show results if Oracle Cloud Infrastructure cloud operations hasn't performed any actions or run any statements in your Autonomous Database instance. See View Oracle Cloud Infrastructure Operations Actions for more information.

September 2022

Feature	Description
Expanded list of Databases with Oracle-Managed Heterogeneous Connectivity	Autonomous Database support for Oracle-managed heterogeneous connectivity makes it easy to create database links to non-Oracle databases. When you use database links with Oracle-managed heterogeneous connectivity, Autonomous Database configures and sets up the connection to the non-Oracle database. The list of supported non-Oracle databases is expanded to include Hive and MongoDB. See Create Database Links to Non-Oracle Databases with Oracle-Managed Heterogeneous Connectivity for more information.
Oracle APEX Upgrade Events	The `APEXUpgradeBegin` and `APEXUpgradeEnd` events are generated when you are using Oracle APEX and your Autonomous Database instance begins an upgrade or completes an upgrade to a new Oracle APEX release. See About Events Based Notification and Automation on Autonomous Database for more information.
Oracle MTS (OraMTS) Recovery Service	Use Oracle MTS (OraMTS) Recovery Service to resolve in-doubt Microsoft Transaction Server transactions on Autonomous Database. See Using OraMTS Recovery Feature on Autonomous Database for more information.

August 2022

Feature	Description
Documentation Addition: Updated Python Connection Information	You can connect Python applications to your Autonomous Database instance either with a wallet (mTLS) or without a wallet (TLS), using the python-oracledb driver. See Connect Python Applications to Autonomous Database for more information.
Autonomous Data Guard SLA	Autonomous Database provides a higher 99.995% availability SLA service commitment with Autonomous Data Guard enabled with a standby database. Database availability SLA is calculated based on the "Monthly Uptime Percentage" described in the Oracle PaaS and IaaS Public Cloud Services Pillar Document, document under **Delivery Policies**.
Oracle Machine Learning for R (OML4R)	Oracle Machine Learning for R (OML4R) is a component of the Oracle Machine Learning family of products, which integrates R with Autonomous Database. Use Oracle Machine Learning for R to: • Perform data exploration and data preparation while seamlessly leveraging Oracle Database as a high-performance computing environment. • Run user-defined R functions on database spawned and controlled R engines, with system-supported data-parallel and task-parallel capabilities. • Access and use powerful in-database machine learning algorithms from R language. See About Oracle Machine Learning for R for more information.

Feature	Description
Synchronization with Data Catalog Enhancements	Use the `DBMS_DCAT$SYNC_LOG` view to easily access the log for the latest Data Catalog synchronization execution. See DBMS_DCAT SYNC_LOG View for more information.
Regional Availability Metrics	You can view regional availability metrics by data center for Oracle Autonomous Database. See Monitor Regional Availability of Autonomous Databases for more information.

July 2022

Feature	Description
Use Oracle Java	Autonomous Database supports Oracle JVM. Oracle JVM is a standard, Java-compatible environment that runs any pure Java application. See Using Oracle Java on Autonomous Database for more information.
Oracle APEX 22.1	Autonomous Database uses Oracle APEX Release 22.1. See Creating Applications with Oracle APEX in Autonomous Database for more information.
Wallet Rotation with a Grace Period	Autonomous Database allows you to rotate wallets for an Autonomous Database instance or for all instances that a cloud account owns in a region, with a grace period of 1 hour to 24 hours. See Rotate Wallets for Autonomous Database for more information.
Change Display Name	You can change the display name for an Autonomous Database instance. See Update the Display Name for an Autonomous Database Instance for more information.
Default Database Name	The default database name and the matching default display name supplied when you provision or clone an instance is a generated 16-character string. See Provision Autonomous Database and Clone an Autonomous Database Instance for more information.
Oracle Real Application Testing: Database Replay	You can use Oracle Real Application Testing Database Replay to capture workload from an on-premises or other cloud service database and replay it on an Autonomous Database instance. This enables you to compare workloads between an on-premises database or other cloud service database and an Autonomous Database. See Using Oracle Real Application Testing - Database Replay for more information.
Export Data with Column Names in Header Row	You can optionally write column names as the first line in output files when you use the `EXPORT_DATA` procedure with CSV output. See EXPORT_DATA Procedure and DBMS_CLOUD Package Format Options for EXPORT_DATA with Text Files (CSV, JSON, and XML for more information.

June 2022

Feature	Description
Character Set Support	The Autonomous Database default database character set is Unicode AL32UTF8 and the default national character set is AL16UTF16. When you provision a database, depending on the workload type, you can select a database character set and a national character set. See Choose a Character Set for Autonomous Database for more information.
Support for Oracle LogMiner	Autonomous Database supports using Oracle LogMiner. See Oracle LogMiner for more information.
Synchronization with Data Catalog Enhancements	Synchronize Data Catalog partitioned metadata with Autonomous Database to create partitioned external tables. See the DBMS_DCAT RUN_SYNC procedure.
Database Actions Changes	Database Actions includes new features. See Changes in Oracle Database Actions for more information.
Network Security Group (NSG) Configuration with Private Endpoint	For a private endpoint, the incoming and outgoing connections are limited by the combination of ingress and egress rules defined in NSGs and in the Security Lists associated with the private endpoint's VCN. Adding an NSG is now optional; when you do not include an NSG the ingress and egress rules defined in the Security Lists for the VCN still apply. See Configuring Network Access with Private Endpoints for more information.
Simplified Configuration Steps with CMU for Microsoft Active Directory	You can configure Autonomous Database to authenticate and authorize Microsoft Active Directory users. The configuration steps are simplified for enabling Autonomous Database with Centrally Managed Users (CMU). See Use Microsoft Active Directory with Autonomous Database for more information.

May 2022

Feature	Description
Azure Active Directory (Azure AD) Integration	Azure Active Directory (Azure AD) users can connect to an Autonomous Database instance using Azure OAuth2 access tokens. See Use Azure Active Directory (Azure AD) with Autonomous Database for more information.
OCPU Limit for License Type Bring Your Own License (BYOL)	The license type Bring Your Own License (BYOL), Oracle Database Standard Edition sets the maximum number of OCPUs that you can use to 8. See View and Update Your License and Oracle Database Edition on Autonomous Database for more information.
Kerberos Authentication Support	You can use Kerberos to authenticate Autonomous Database users. See Configure Kerberos Authentication for more information.
Autonomous Data Guard Standby State	Autonomous Data Guard shows a standby database in "Standby" state. See Autonomous Database Standby Database State for more information.

Feature	Description
Database Actions Additions	Database Actions provides all the functionality that is available on the Autonomous Database Service Console. The Service Console will be deprecated soon. See Service Console Replacement with Database Actions for more information on where to find Service Console functionality in Database Actions.
Autonomous Data Guard Support for Terraform	Autonomous Data Guard supports operation from Terraform scripts See the following for more information: • Use the API • oci_database_autonomous_database • Oracle Cloud Infrastructure Provider • Terraform Registry Data Source: Data Source: oci_database_autonomous_database
Use Azure Service Principal	You can use an Azure service principal with Autonomous Database to access Azure resources without having to create and store your own credential objects in the database. See Use Azure Service Principal to Access Azure Resources for more information.
Monitor Autonomous Database Availability	You can monitor availability information for your Autonomous Database instance form the Oracle Cloud Infrastructure Console or by viewing metrics. See Monitor Autonomous Database Availability and View Metrics for an Autonomous Database Instance for more information.
Database Names Length Limit	The length limit for database names when you create or clone your database has been increased from 14 to 30 characters. This change provides more flexibility, giving you the option to create longer database names. See Provision Autonomous Database or Clone an Autonomous Database Instance for more information.
Egress Rules for all Outbound Connections with Private Endpoints	When you define a private endpoint for your Autonomous Database instance you can provide enhanced security by setting a database property to enforce that all outgoing connections to a target host are subject to and limited by the private endpoint's egress rules. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

April 2022

Feature	Description
Display Autonomous Database Availability	You can monitor Autonomous Database availability from the Oracle Cloud Infrastructure Console or using metrics. See Monitor Autonomous Database Availability for more information.
OML4Py SQL API for Embedded Python Execution	OML4Py embedded Python execution on Autonomous Database supports a SQL API in addition to the REST API. Embedded execution allows you to run user-defined Python functions in database-spawned and managed Python engines, along with data-parallel and task-parallel automation. See SQL API for Embedded Python Execution with On-premises Database for more information.

Feature	Description
Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint without a Wallet	You can create database links from an Autonomous Database to a target Oracle Database that is on a private endpoint and connect without a wallet (TCP). See Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint without a Wallet for more information.
Availability Service Level Objectives (SLOs)	Describes the Service Level Objectives (SLOs) for Oracle Autonomous Database. See Availability Service Level Objectives (SLOs) for more information.
Synchronization with Data Catalog Enhancements	The `DBMS_DCAT RUN_SYNC` procedure has new parameters to give you more control over what objects are synchronized and how the synchronization is performed. See CREATE_SYNC_JOB Procedure and RUN_SYNC Procedure for more information.
Data Catalog Views	Discover all accessible data catalogs in the current region or across all regions. See ALL_DCAT_GLOBAL_ACCESSIBLE_CATALOGS View and ALL_DCAT_LOCAL_ACCESSIBLE_CATALOGS View for more information.

March 2022

Feature	Description
Oracle APEX 21.2	Autonomous Database uses Oracle APEX Release 21.2. See Creating Applications with Oracle APEX in Autonomous Database for more information.
OML4Py Client Access	A new client package for OML4Py provides a “universal" client in that users can access Autonomous Database from a standalone Python engine on Linux to connect to Autonomous Database as well as 19c and newer Oracle Database releases. This supports working with OML4Py on Autonomous Database, local third-party IDEs, and other notebook environments, like JupyterLab. It further allows switching between on-premises and Autonomous Database instances using the same client package. See Install OML4Py Client for Linux for Use With Autonomous Database for more information.
Oracle APEX Upgrade Available Event	The `APEXUpgradeAvailable` event is generated when you are using Oracle APEX and a new release becomes available. See About Events Based Notification and Automation on Autonomous Database for more information.
Database Links with Oracle-Managed Heterogeneous Connectivity	Autonomous Database support for Oracle-managed heterogeneous connectivity makes it easy to create database links to non-Oracle databases. When you use database links with Oracle-managed heterogeneous connectivity, Autonomous Database configures and sets up the connection to the non-Oracle database. See Create Database Links to Non-Oracle Databases with Oracle-Managed Heterogeneous Connectivity for more information.
Metadata Columns in External Tables	To identify which file each row is coming from in your external tables, you can query the columns named `file$name` and `file$path` to find the source file name and the Object Store path URL for that file. See External Table Metadata Columns for more information.

Feature	Description
External Tables with Partitioning Specified in Source Files	When you use external file partitioning with the partitions specified in external source files, Autonomous Database analyzes Cloud Object Store path information to determine the partition columns and data types. See Query External Tables with Partitioning Specified in Source Files for more information.
Identity and Access Management (IAM) Authentication Additional Features	You can configure Autonomous Database to use Oracle Cloud Infrastructure Identity and Access Management (IAM) authentication and authorization to allow IAM users to access an Autonomous Database with IAM credentials. New additions for IAM support include the following: defining global user mapping, defining global roles mapping, support for resource principal usage with IAM, and proxy user support. See Use Identity and Access Management (IAM) Authentication with Autonomous Database for more information.
Storage Auto Scaling	With **Storage auto scaling** enabled the Autonomous Database can expand to use up to three times the reserved base storage. If you need additional storage, the database automatically uses the reserved storage without any manual intervention required. Storage auto scaling is disabled by default. See Storage Auto Scaling for more information.
Documentation Addition: Get Help, Search Forums, and Contact Support	When you use Autonomous Database, sometimes you need to get help from the community or to talk to someone in Oracle support. This documentation addition provides information about getting help by viewing and posting questions on forums and using Oracle Cloud Support to create a support request. See Get Help, Search Forums, and Contact Support for more information.

February 2022

Feature	Description
Track Oracle Cloud Infrastructure Resources, Cost and Usage	Autonomous Database provides details on Oracle Cloud Infrastructure resources, cost and usage. See Track Oracle Cloud Infrastructure Resources, Cost and Usage Reports with Autonomous Database Views for more information.
Critical Events with Customer-Managed Keys	Autonomous Database events are triggered when an instance is using customer-managed keys and the database becomes inaccessible or when the database recovers from being inaccessible. See About Critical Events on Autonomous Database for more information.
Time Zone Handling in Calls to `SYSDATE` and `SYSTIMESTAMP`	The initialization parameter `SYSDATE_AT_DBTIMEZONE` parameter enables special handling in a session for the date and time value returned in calls to `SYSDATE` and `SYSTIMESTAMP`. Depending on the value of `SYSDATE_AT_DBTIMEZONE`, you see either the date and time based on the default Autonomous Database time zone, Coordinated Universal Time (UTC), or based on the time zone that you set in your database. See SYSDATE_AT_DBTIMEZONE for more information.

January 2022

Feature	Description
Use a Cloud Code Repository	Autonomous Database provides routines to manage and store files in Cloud Code (Git) Repositories. The supported Cloud Code Repositories are: GitHub Repository, AWS CodeCommit, and Azure Repos. See Using and Managing a Cloud Code Repository with Autonomous Database for more information.
Oracle Machine Learning Notebooks Repository	Oracle Machine Learning Notebooks repositories are stored in a schema on the Autonomous Database instance. This enables database instance-specific backup and restore of Oracle Machine Learning Notebooks as well as support for cross-region Autonomous Data Guard. See What's New in Oracle Machine Learning on Autonomous Database for more information.
Oracle Machine Learning Notebooks Import and Export of Jupyter Format Notebooks	Oracle Machine Learning Notebooks supports importing and exporting Jupyter format notebooks. This makes it easier for users to interoperate with other notebook environments. See Export a Notebook and Import a Notebook for more information.
Add My IP Address Button	When you provision an Autonomous Database instance and configure network access with Access Control Lists (ACLs) or when you update network access, click **Add My IP Address** to add your current IP address to the ACL entry. See Configuring Network Access with Access Control Rules (ACLs) for more information.
Reconnect a Disconnected Refreshable Clone	You can reconnect a refreshable clone to the source database. This allows you to use a refreshable clone as a test database and run DML and make changes while the database is disconnected. When you are done with your testing you can reconnect to the source database, which refreshes the clone to the point where it was when you disconnected. See Reconnect a Refreshable Clone to the Source Database for more information.
GitHub Raw URL support in `DBMS_CLOUD`	Use GitHub Raw URLs with `DBMS_CLOUD` APIs to access source files that reside on a GitHub Repository. See GitHub Raw URL Format for more information.
View Maintenance Status and Timezone Version Notifications	The `DB_NOTIFICATIONS` view stores information about maintenance status notifications and timezone version upgrade notifications for your Autonomous Database instance. See View Maintenance Status and Timezone Version Notifications for more information.
One-way TLS Connections on Linux x64 Clients using Oracle Call Interface without a Wallet	Using TLS authentication for Oracle Call Interface connections on Linux x64, a wallet is not required when the client program connects with Oracle Instant Client 19.13. Using TLS authentication for Oracle Call Interface connections on all other platforms, or without Oracle Instant Client 19.13, clients must provide a generic CA root certificate wallet. See Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections Using TLS Authentication for more information.

2021 What's New

Announcements for the noteworthy changes made to Oracle Autonomous Database Serverless Deployment in the year 2021.

December 2021

Feature	Description
Oracle Database API for MongoDB	Oracle Database API for MongoDB lets applications interact with collections of JSON documents in Autonomous Database using MongoDB commands. See Using Oracle Database API for MongoDB for more information.
Oracle Data Miner Access to Autonomous Database	Oracle Data Miner in SQL Developer can be used with Oracle Autonomous Database. Oracle Data Miner is a SQL Developer extension that enables users to create, schedule, and deploy analytical workflows through a drag-and-drop user interface. Oracle Data Miner serves as a productivity tool for data scientists and an enabler for citizen data scientists with a no-code machine learning environment that automates some common machine learning steps. See Oracle Data Miner for more information.
Autonomous Data Guard Paired Regions	The Autonomous Data Guard paired region list is expanded with additional regions. Autonomous Data Guard paired regions are remote regions where you can create a cross-region standby database. See Autonomous Data Guard Paired Regions for more information.
`DBMS_PIPE` PL/SQL Package	The `DBMS_PIPE` package that lets two or more sessions in the same instance communicate is available in Autonomous Database.
Cross-Region Cloning	When you clone an Autonomous Database instance using the Oracle Cloud Infrastructure Console, you can select your preferred region for the clone. You can create the clone in the current region or in a remote region. The list of available regions to create a clone only shows the regions that you are subscribed to. See Clone an Autonomous Database Instance for more information.
Identity and Access Management (IAM) Authentication	You can configure Autonomous Database to use Oracle Cloud Infrastructure Identity and Access Management (IAM) authentication and authorization to allow IAM users to access an Autonomous Database with IAM credentials. See Use Identity and Access Management (IAM) Authentication with Autonomous Database for more information.

Feature	Description
Schedule Start and Stop Times	When you enable an Auto Start/Stop Schedule for an Autonomous Database instance, the instance automatically starts and stops according to the schedule you specify. This allows you to reduce costs by scheduling shutdown periods for times when a system is not in use. See Schedule Start and Stop Times for an Autonomous Database Instance for more information.
PL/SQL API for Switching Consumer Groups	Use the `CS_SESSION` Package to switch the database service and consumer group of an existing session. See CS_SESSION Package for more information.
Export Data as CSV, JSON, or XML Data Files	You can export data from Autonomous Database as text in the following formats: CSV, JSON, or XML. See Move Data to Object Store as CSV, JSON, or XML Using EXPORT_DATA for more information.
Use Database Links with a Target Database on a Private Endpoint	Create database links from an Autonomous Database source to a target Oracle Database that is on a private endpoint. To create a database link to a target database on a private endpoint, the target database must be accessible from the source database's Oracle Cloud Infrastructure VCN. See Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint for more information.

November 2021

Feature	Description
Database Management	The **Associated Services** area on the Oracle Cloud Infrastructure Console includes a **Database Management** link. You can use Database Management service to monitor the health of a single Autonomous Database or a fleet of Autonomous Databases. See Use Database Management Service to Monitor Databases for more information.
Database Actions Access	Access Database Actions on the Oracle Cloud Infrastructure Console with **Database Actions** button. See Connect with Built-in Oracle Database Actions for more information.
DBMS_LDAP and UTL_TCP Packages	The DBMS_LDAP and UTL_TCP packages are available, with some restrictions. See Restrictions and Notes for Database PL/SQL Packages for more information.
Concurrent Operations with Lifecycle Management Actions	When you initiate operations that take some time to complete, including scaling the system (Scale Up/Down), these operations no longer prevent you from performing other operations. For example, if you perform certain database lifecycle management actions such as stopping the database during a long-running operation, depending on the operation, the long running operation is either canceled or paused. See Concurrent Operations on Autonomous Database for more information.

October 2021

Feature	Description
Import JSON Data from Cloud Object Store	You can import JSON data from Cloud Object Store into a table. This import method supports all the Cloud Object Stores supported by Autonomous Database, and you can use an Oracle Cloud Infrastructure resource principal to access your Oracle Cloud Infrastructure Object Store or Amazon Resource Names (ARNs) to access AWS Simple Storage Service (S3). See Create Credentials and Copy JSON Data into an Existing Table for more information.
Blockchain Tables Support	Blockchain tables are insert-only tables that organize rows into a number of chains. Blockchain tables protect data that records important actions, assets, entities, and documents from unauthorized modification or deletion by criminals, hackers, and fraud. Blockchain tables prevent unauthorized changes made using the database and detect unauthorized changes that bypass the database. See Managing Blockchain Tables for more information.
Immutable Tables Support	Immutable tables are read-only tables that prevent unauthorized data modifications by insiders and accidental data modifications resulting from human errors. See Managing Immutable Tables for more information.
Integrate with Oracle Cloud Infrastructure Data Catalog	OCI Data Catalog metadata is synchronized with Autonomous Database creating schemas and external tables. You are allowed to immediately query data available in object store. See Query External Data with Data Catalog for more information.
Oracle APEX 21.1	Autonomous Database uses Oracle APEX Release 21.1. See Creating Applications with Oracle Application Express in Autonomous Database for more information.

September 2021

Feature	Description
TLS and Mutual TLS Connections	Autonomous Database by default supports Mutual TLS (mTLS) connections. You have the option to configure an Autonomous Database instance to support both mTLS and TLS connections. See the following for more information: • About Connecting to an Autonomous Database Instance • Update Network Options to Allow TLS or Require Only Mutual TLS (mTLS) Authentication on Autonomous Database
Vanity URLs for APEX, ORDSand Database Tools	You can configure a custom domain name or vanity URL for your APEX apps, Oracle REST Data Services (ORDS), and developer tools by placing an Oracle Cloud Infrastructure Load Balancer directly in front of your Autonomous Database. See the following for additional details, including prerequisites and step-by-step instructions: • Access Oracle APEX, Oracle REST Data Services, and Developer Tools Using a Vanity URL • Access Oracle REST Data Services, Oracle APEX, and Developer Tools Using a Vanity URL

August 2021

Feature	Description
Cross-Region Autonomous Data Guard	Autonomous Data Guard allows you to enable a cross-region standby (peer) database to provide data protection and disaster recovery for your Autonomous Database instance. When you enable Autonomous Data Guard, the system creates a standby database that is continuously updated with the changes from the primary database. You can enable Autonomous Data Guard with a standby in the current region, a local standby, or with a standby in a different region, a cross-region standby. You can also enable Autonomous Data Guard with both a local standby and a cross-region standby. See Using Standby Databases with Autonomous Database for Disaster Recovery for more information.
SQL Tracing for Database Session	Use SQL tracing to help you identify the source of an excessive database workload, such as a high load SQL statement in your application. See Perform SQL Tracing on Autonomous Database for more information.
Export Data as JSON to Cloud Object Store	Use the procedure `DBMS_CLOUD.EXPORT_DATA` to export data to your Cloud Object Store as JSON, by specifying a query. This export method supports all the Cloud Object Stores supported by Autonomous Database, and you can use an Oracle Cloud Infrastructure resource principal to access your Oracle Cloud Infrastructure Object Store or Amazon Resource Names (ARNs) to access AWS Simple Storage Service (S3). See Move Data to Object Store as JSON Data Using EXPORT_DATA for more information.
Documentation Addition: Load Data from a Source File with Fixed Width Data	The documentation provides an example for loading data from a fixed-width source file to an external table. See Example Loading Data from a Fixed Width File for more information.
Set Patch Level	When you provision or clone an Autonomous Database instance you can select a patch level to apply upcoming patches. There are two patch level options: Regular and Early. After your provision or clone an Autonomous Database instance and set the patch level to Early, maintenance windows for the instance are scheduled and applied one week before instances with the patch level set to Regular. The Early patch level allows you to use and to test upcoming patches before they are applied to all systems. See Set the Patch Level for more information.

July 2021

Feature	Description
View Patch Details	You can view Autonomous Database patch information, including a list of resolved issues. See View Patch Information for more information.

Feature	Description
Include Developer Tools Links in Wallet Readme File	The wallet `README` file includes links for Autonomous Database tools and resources, including: Database Actions, Graph Studio, Oracle APEX, Oracle Machine Learning Notebooks, Autonomous Database Service Console, and SODA drivers. See Wallet README File for more information.

June 2021

Feature	Description
Change MEDIUM Service Concurrency Limit from Service Console	If your application requires a customized concurrency limit not available with the predefined services, you can modify the concurrency limit for the MEDIUM service from the Autonomous Database Service Console or using PL/SQL procedures. See Change MEDIUM Service Concurrency Limit for more information.
Automatic Partitioning	Automatic partitioning analyzes and automates partition creation for tables and indexes of a specified schema to improve performance and manageability in Autonomous Database. Automatic partitioning, when applied, is transparent and does not require user interaction or maintenance. Automatic partitioning does not interfere with existing partitioning strategies and manually partitioned tables are excluded as candidates for automatic partitioning. See Manage Automatic Partitioning on Autonomous Database for more information.
Use Customer-Managed Encryption Keys	Autonomous Database provides two options for Transparent Data Encryption (TDE) to encrypt data in the database: • Oracle-managed encryption keys • Customer-managed encryption keys See Managing Encryption Keys on Autonomous Database for more information.
Autonomous Database RMAN Recovery Catalog	You can use Autonomous Database as a Recovery Manager (RMAN) recovery catalog. A recovery catalog is a database schema that RMAN uses to store metadata about one or more Oracle databases. See Autonomous Database RMAN Recovery Catalog for more information.
Read-Only Mode for Sessions	You can set the Autonomous Database operation mode to read-only for a session. In read-only mode users for the session can only run queries. See Change Autonomous Database Operation Mode for a Session for more information.
Documentation Addition: `CS_RESOURCE_MANAGER` Package	The `CS_RESOURCE_MANAGER` package provides procedures to list and update consumer group parameters, and to revert parameters to default values. See CS_RESOURCE_MANAGER Package for more information.

May 2021

Feature	Description
Transparent Application Continuity (TAC) on Autonomous Database	Autonomous Database provides application continuity features for making connections to the database. You enable Application Continuity on Autonomous Database in one of two configurations: Application Continuity(AC) or Transparent Application Continuity (TAC). Application Continuity hides outages for thin Java-based applications, Oracle Database Oracle Call Interface, and ODP.NET based applications with support for open-source drivers, such as Node.js, and Python. Transparent Application Continuity (TAC) transparently tracks and records session and transactional state so the database session can be recovered following recoverable outages. This is done with no reliance on application knowledge or application code changes. See Using Application Continuity on Autonomous Database for more information.
Amazon Resource Names (ARNs) to Access AWS Resources	You can use Amazon Resource Names (ARNs) to access AWS resources with Autonomous Database. When you use ARN role based authentication with Autonomous Database, you can securely access AWS resources without creating and saving credentials based on long-term AWS IAM access keys. See Use Amazon Resource Names (ARNs) to Access AWS Resources for more information.
Resource Principal to Access Oracle Cloud Infrastructure Resources	When you use a resource principal with Autonomous Database, you or your tenancy administrator define the Oracle Cloud Infrastructure policies in a dynamic group that allows you to access resources. With a resource principal you do not need to create a credential object and Autonomous Database creates and secures the resource principal credentials you use to access the Oracle Cloud Infrastructure resources specified in the dynamic group. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.
Graph Studio	Graph Studio features include automated modeling to create graphs from database tables, an integrated notebook to run graph queries and analytics, and native graph and other visualizations. You can invoke nearly 60 pre-built graph algorithms and visualize your data with many visualization options. Graph Studio is a fully integrated, automated feature with Autonomous Database. See Using Oracle Graph with Autonomous Database for more information.

April 2021

Feature	Description
Change MEDIUM Service Concurrency Limit	If your application requires a customized concurrency limit not available with the predefined services, you can modify the concurrency limit for the MEDIUM service. See Change MEDIUM Service Concurrency Limit for more information.

Feature	Description
DBMS_CLOUD REST API Results Cache	The DBMS_CLOUD REST API functions allow you to make HTTP requests and obtain and save results. Saving results in the cache allows you to view past results with the `SESSION_CLOUD_API_RESULTS` view. Saving and querying historical results of `DBMS_CLOUD` REST API requests can help you when you need to work with previous results in your applications. See DBMS_CLOUD REST API Results Cache for more information.
View and Manage Customer Contacts for Operational Issues and Announcements	When customer contacts are set, Oracle sends notifications to the specified email addresses to inform you of service-related issues. Contacts in the customer contacts list receive unplanned maintenance notices and other notices, including but not limited to notices for database upgrades and upcoming wallet expiration. See View and Manage Customer Contacts for Operational Issues and Announcements for more information.
Always Free Autonomous JSON Database	You have the option to create a limited number of Always Free Autonomous JSON Databases that do not consume cloud credits. Always Free Autonomous JSON Databases can be created in Oracle Cloud Infrastructure accounts that are in a trial period, have paying status, or are always free. See Always Free Autonomous Database for more information.
Always Free Oracle APEX Application Development	You have the option to create a limited number of Always Free APEX Services that do not consume cloud credits. Always Free APEX Service can be created in Oracle Cloud Infrastructure accounts that are in a trial period, have paying status, or are always free. See Always Free Oracle APEX Application Development for more information.
Expiring Wallets Notification	Starting six weeks before the wallet expiration date Oracle sends notification emails each week, indicating the wallet expiration date. These emails provide notice before your wallet expires that you need to download a new wallet. You can also use the `WalletExpirationWarning` event to be notified when a wallet is due to expire. See Download Client Credentials (Wallets) for more information.
Documentation Addition: Oracle Graph	Oracle Graph with Autonomous Database enables you to create graphs from data in your Autonomous Database. With graphs you can analyze your data based on connections and relationships between data entities. See Using Oracle Graph with Autonomous Database for more information.
Documentation Addition: Oracle Spatial	Oracle Spatial with Autonomous Database allows developers and analysts to get started easily with location intelligence analytics and mapping services. See Using Oracle Spatial with Autonomous Database for more information.
Download Data Pump Dump File Set with Script	To support using Oracle Data Pump Import to import a dump file set to a target database, you can use a script that supports substitution characters to download all the dump files from your Object Store in a single command. See Download Dump Files, Run Data Pump Import, and Clean Up Object Store for more information.

March 2021

Feature	Description
Autonomous Database Tools for Database Actions	Autonomous Database provides the following data tools: • **Data Load**: helps you select data to load from your local computer, from tables in other databases, or from cloud storage. Then you can add the data you select to new or existing tables or views in Autonomous Database. See The Data Load Page for more information. • **Catalog**: helps you to see information about the entities in your Autonomous Database, and to see the effect that changing an object has on other objects. Catalog provides a tool for you to examine data lineage and understand the impact of changes. See The Catalog Page for more information. • **Data Insights**: crawls a table or business model and discovers hidden patterns, anomalies, and outliers in your data. Insights are automatically generated by built-in analytic functions. The results of the insight analysis appear as bar charts in the Data Insights dashboard. See The Data Insights Page for more information. • **Business Models**: describes the business entities that are derived from data in your Autonomous Database schema or from other sources. This allows you to create a semantic model on top of your data identifying hierarchies, measures, and dimensions. See The Business Models Page for more information.
Maintenance History	You can view Autonomous Database maintenance history to see details about past maintenance, such as the start time and stop time for past maintenance events. See About Autonomous Database Maintenance Windows and History for more information.
Oracle Machine Learning AutoML User Interface	AutoML User Interface (AutoML UI) is an Oracle Machine Learning interface that provides you with no-code automated machine learning. When you create and run an experiment in AutoML UI, it automatically performs algorithm and feature selection, as well as model tuning and selection, thereby enhancing productivity as well as model accuracy and performance. See Get Started with AutoML for more information.
Oracle Machine Learning Models	REST endpoints are available so that you can store Machine Learning models and create scoring endpoints for the models. The REST API for Oracle Machine Learning Services supports both Oracle Machine Learning models and ONNX format models. See REST API for Oracle Machine Learning Services for more information.
ADMIN Password Expiration Warning Event	The `AdminPasswordWarning` event produces an event when the Autonomous Database ADMIN password is expiring within 30 days or is expired. See About Events Based Notification and Automation on Autonomous Database for more information.

Feature	Description
Oracle Machine Learning for Python (OML4Py)	Oracle Machine Learning for Python (OML4Py) on Autonomous Database provides a Python API to explore and prepare data, and build and deploy machine learning models using the database as a high-performance compute engine. OML4Py is available through the Python interpreter in Oracle Machine Learning Notebooks. See Get Started with Notebooks for Data Analysis and Data Visualization for more information.

February 2021

Feature	Description
Oracle Database Actions	SQL Developer Web is now called Database Actions. Database Actions is a web-based interface that provides development tools, data tools, administration, and monitoring features for Autonomous Database. Using Database Actions you can load data and run SQL statements, queries, and scripts in a worksheet. See Connect with Built-in Oracle Database Actions for more information.
Query Big Data Service Hadoop (HDFS) Data	Big Data Service provides enterprise-grade Hadoop as a service, with end-to-end security, high performance, and ease of management and upgradeability. After deploying the Oracle Cloud SQL Query Server to Big Data Service, you can easily query data available on Hadoop clusters at scale from Autonomous Database using SQL. See Query Big Data Service Hadoop (HDFS) Data from Autonomous Database for more information.

January 2021

Feature	Description
Autonomous Database Events	Autonomous Database generates events that you can subscribe to with Oracle Cloud Infrastructure Events. There are two new Information events: `ScheduledMaintenanceWarning` and `WalletExpirationWarning`. See Use Autonomous Database Events for more information.
Private Endpoint Support for Tools (Oracle Machine Learning Notebooks are supported with Private Endpoints)	You can access Oracle APEX, Oracle SQL Developer Web, Oracle REST Data Services, and Oracle Machine Learning Notebooks with Private Endpoints. See Configuring Network Access with Private Endpoints for more information.

2020 What's New

Announcements for the noteworthy changes made to Oracle Autonomous Database Serverless Deployment in the year 2020.

- December 2020
- November 2020
- October 2020
- September 2020

- August 2020
- July 2020
- June 2020
- May 2020
- April 2020
- March 2020
- February 2020
- January 2020

December 2020

Feature	Description
Replicate Data and Capture Data with Oracle GoldenGate	Using Oracle GoldenGate you can capture changes from an Oracle Autonomous Database and replicate to any target database or platform that Oracle GoldenGate supports, including another Oracle Autonomous Database. See Oracle GoldenGate Capture for Oracle Autonomous Database for more information.
Oracle APEX Application Development	Oracle APEX Application Development (APEX Service) is a low cost Oracle Cloud service offering convenient access to the Oracle Application Express platform for rapidly building and deploying low-code applications. See Oracle APEX Application Development for more information.
Oracle Database 21c with Always Free Autonomous Database	Oracle Database 21c is available with Always Free Autonomous Database. See the following for more information: • Always Free Autonomous Database • Always Free Autonomous Database Oracle Database 21c Features
Oracle APEX 20.2	Autonomous Database uses Oracle APEX 20.2. See Creating Applications with Oracle Application Express in Autonomous Database for more information.

November 2020

Feature	Description
Amazon S3 Presigned URLs	You can use a presigned URL in any `DBMS_CLOUD` procedure that takes a URL to access files in Amazon Simple Storage Service, without the need to create a credential. See Amazon S3 Compatible URI Format for more information.
Azure Blob Storage Shared Access Signatures (SAS) URLs	You can use Shared Access Signatures (SAS) URL in any `DBMS_CLOUD` procedure that takes a URL to access files in Azure Blob Storage, without the need to create a credential. See Azure Blob Storage URI Format for more information.

October 2020

Feature	Description
User-Defined Profiles Minimum Password Length	If you create a user-defined profile with a Password Verification Function (PVF), then the minimum password length can be 8 characters. See Manage Password Complexity on Autonomous Database for more information.
PL/SQL SDK for Oracle Cloud Infrastructure Resources	PL/SQL SDK for Oracle Cloud Infrastructure enables you to write code to manage Oracle Cloud Infrastructure resources. The PL/SQL SDK is on all Autonomous Database Serverless products. See PL/SQL SDK for more information.
Documentation Addition: Oracle Database Actions User Management	You can quickly create users and modify account settings for users with Oracle Database Actions. See Create Users on Autonomous Database for more information.
Autonomous Database Events	Autonomous Database generates events that you can subscribe to with Oracle Cloud Infrastructure Events. Subscribing to Autonomous Database events allows you to create automation and to receive notifications when the events occur. See Use Autonomous Database Events for more information.

September 2020

Feature	Description
Private Endpoint Support for Tools	You can access Oracle APEX, Oracle SQL Developer Web, and Oracle REST Data Services with Private Endpoints. See Configuring Network Access with Private Endpoints for more information.
Read-Only and Restricted Modes	You can select an Autonomous Database operation mode. The default mode is Read/Write. If you select Read-Only mode users can only run queries. In addition, for either mode you can restrict access to only allow privileged users to connect to the database. See Change Autonomous Database Mode to Read/Write or Read-Only for more information.
Refreshable Clones	Autonomous Database provides cloning where you can choose to create a full clone of the active instance, create a metadata clone, or create a refreshable clone. With a refreshable clone the system creates a clone that can be easily updated with changes from the source database. See Using Refreshable Clones with Autonomous Database for more information.
Documentation Addition: Auditing Autonomous Database	This new chapter describes the auditing features that allow you to track, monitor, and record activity on your database. Auditing can help you detect security risks and improve regulatory compliance for your database. See Auditing Autonomous Database for more information.

August 2020

Feature	Description
Autonomous JSON Database	If your database version is Oracle Database 19c or higher, you can provision an Autonomous JSON Database. Autonomous JSON Database is an Oracle Cloud service that is specialized for developing NoSQL-style applications that use JavaScript Object Notation (JSON) documents. See About Autonomous JSON Database for more information.
Amazon S3-Compatible Object Stores	Autonomous Database supports Amazon S3-Compatible object stores, such as Oracle Cloud Infrastructure Object Storage, Google Cloud Storage, and Wasabi Hot Cloud Storage. See Load Data from Files in the Cloud for more information.
Asynchronous Requests using `SEND_REQUEST`	The `DBMS_CLOUD SEND_REQUEST` function supports long running requests with the additional parameters: `async_request_url`, `wait_for_states`, and `timeout`. See SEND_REQUEST Function for more information.
Rename Database	If your database version is Oracle Database 19c or higher, you can rename your Autonomous Database. See Rename Autonomous Database for more information.

July 2020

Feature	Description
Oracle APEX Release 20.1	Autonomous Database supports Oracle APEX Release 20.1. See Creating Applications with Oracle Application Express in Autonomous Database for more information. If your Autonomous Database instance is on Oracle Database 18c, to use APEX 20.1 you must upgrade to Oracle Database 19c. After you upgrade to Oracle Database 19c, APEX is automatically upgraded to APEX 20.1.
Use a Standby Database	Autonomous Database provides the Autonomous Data Guard feature to enable a standby (peer) database to provide data protection and disaster recovery for your Autonomous Database instance. See Using a Standby Database with Autonomous Database for more information.
Create and Alter User Profiles	You can create and alter database user profiles. See Manage User Profiles with Autonomous Database for more information.
Manage Time Zone File Versions	The time zone files are periodically updated to reflect the latest time zone specific changes. Autonomous Database automatically picks up the updated time zone files. See Manage Time Zone File Version on Autonomous Database for more information.

Feature	Description
Network Access Changes	If your Autonomous Database network access is configured to use a private endpoint you can change the configuration to use a public endpoint. Likewise, if your Autonomous Database instance is configured to use a public endpoint, you can change the configuration to use a private endpoint. Autonomous Database network access changes from a public to a private endpoint, or from a private to a public endpoint are only supported with database versions Oracle Database 19c onwards. See Configuring Network Access with Access Control Rules (ACLs) and Private Endpoints for more information.
Use Database Links to Access Non-Oracle Databases	You can create database links from an Autonomous Database to an Oracle Database Gateway to access Non-Oracle databases. See Create Database Links to an Oracle Database Gateway to Access Non-Oracle Databases for more information.

June 2020

Feature	Description
ORC Format and Complex Types	Autonomous Database supports loading and querying data in object store in ORC format, in addition to Avro and Parquet. Also, for ORC, Avro, and Parquet structured file types you can load and query complex data types. Support for ORC format and complex types requires Oracle Database 19c. See Load Data from Files in the Cloud and Query External Data with ORC, Parquet, or Avro Source Files for more information.
`PUT_OBJECT` Maximum File Transfer Size Increase	The `DBMS_CLOUD.PUT_OBJECT` procedure maximum size limit for file transfers to Oracle Cloud Infrastructure Object Storage is increased to 50 GB. See PUT_OBJECT Procedure for more information.
Customer Managed Oracle REST Data Services	You can use a customer managed environment to run Oracle REST Data Services (ORDS) if you want to control the configuration and management of ORDS. Installing and configuring a customer managed environment for ORDS allows you to run ORDS with configuration options that are not possible using the default Oracle managed ORDS available with Autonomous Database. See About Customer Managed Oracle REST Data Services on Autonomous Database for more information.

May 2020

Feature	Description
Move Selective Data to Object Store	You can export data to Oracle Data Pump dump files on Object Store by specifying a query to select the data to export. See Move Data to Object Store as Oracle Data Pump Files Using EXPORT_DATA for more information.

Feature	Description
Obtain Tenancy Details for a Service Request	When you file a service request for Autonomous Database, you need to provide the tenancy details for your instance. Tenancy details for the instance are available on the Oracle Cloud Infrastructure console or you can obtain these details by querying the database. See Obtain Tenancy Details to File a Service Request for more information.
Upgrade Autonomous Database to Oracle Database 19c	If your Autonomous Database instance is on Oracle Database 18c, you can upgrade to Oracle Database 19c by clicking **Upgrade to 19c**.

April 2020

Feature	Description
SODA Documents and Collections	Autonomous Database supports loading and using Simple Oracle Document Architecture (SODA) documents and collections. SODA allows you to store, search, and retrieve document collections, typically JSON documents, without using SQL. You can also access document collections with SQL/JSON operators, providing you with the simplicity of a document database but with the power of a relational database. See Using JSON Documents with Autonomous Database for more information.
Load Data with Oracle Data Pump Access Driver Dump Files	You can use Oracle Data Pump dump files in the Cloud as source files to load your data. The files for this load type must be exported from the source system using the `ORACLE_DATAPUMP` access driver in External Tables. See Create Credentials and Load Data Pump Dump Files into an Existing Table for more information.
Create External Tables and Query Data with Oracle Data Pump Access Driver Dump Files	You can query Oracle Data Pump dump files in the Cloud by creating an external table. The source files to create this type of external table must be exported from the source system using the `ORACLE_DATAPUMP` access driver in External Tables. See Query External Data Pump Dump Files for more information.
Oracle Extensions for IDEs	You can use Oracle extensions for IDEs to develop applications on Autonomous Database. Extensions are available for Eclipse, Microsoft Visual Studio, and Microsoft Visual Studio Code (VS Code). These extensions enable you to connect to, browse, and manage Autonomous Databases in Oracle Cloud directly from the IDE. See Use Oracle Extensions for IDEs to Develop Applications for more information.
Per-Second Billing	Autonomous Database instance CPU and storage usage is billed by the second, with a minimum usage period of 1 minute. Previously Autonomous Database billed in one-hour minimum increments and partial usage of an hour was rounded up to a full hour.
Stateful Rule Support in Private Endpoints	When you configure a private endpoint you define a security rule(s) in a Network Security Group (NSG); this creates a virtual firewall for your Autonomous Database and allows connections to the Autonomous Database instance. The NSG can now be configured with stateful rules. See Configure Private Endpoints with Autonomous Database for more information.

Feature	Description
Wallet zip File Contains `README` File	The wallet file that contains client credentials (wallet files) for your Autonomous Database instance now also has a `README` file with wallet expiration information. Wallet files that were downloaded prior to April 2020 do not contain this file. See Download Client Credentials (Wallets) for more information.

March 2020

Feature	Description
Restart Autonomous Database instance	Use **Restart** to restart an Autonomous Database instance. See Restart Autonomous Database for more information.
Oracle Data Pump Direct Export to Object Store	Depending on your cloud object store, you can use Oracle Data Pump to directly export to your object store to move data between Autonomous Database and other Oracle databases. See Move Data with Data Pump Export to Object Store for more information.
Oracle Data Pump Pre-authenticated URLs for Dump Files	If your source files reside on Oracle Cloud Infrastructure Object Storage, you can use pre-authenticated URLs with Oracle Data Pump. See Import Data Using Oracle Data Pump on Autonomous Database for more information.
Documentation Addition: Upgrade Autonomous Database to Oracle Database 19c	Provides information on upgrading to Oracle Database 19c if your Autonomous Database instance is on Oracle Database 18c.
Oracle Database Versions by Region	Depending on the region where you provision or clone your database, Autonomous Database supports one or more Oracle Database versions. Oracle Database 19c is available in all commercial regions.
Oracle Application Express Release 19.2	Autonomous Database supports Oracle Application Express Release 19.2. See Creating Applications with Oracle Application Express in Autonomous Database for more information.

February 2020

Feature	Description
Private Endpoints	This configuration option assigns a private endpoint, private IP and hostname, to a database and allows traffic only from the VCN you specify; access to the database from all public IPs or VCNs is blocked. This allows you to define security rules at the Network Security Group (NSG) level and to control traffic to your Autonomous Database. See About Network Access Options for more information.
Oracle Database Vault	Oracle Database Vault implements powerful security controls for your database. These unique security controls restrict access to application data by privileged database users, reducing the risk of insider and outside threats and addressing common compliance requirements. See Using Oracle Database Vault with Autonomous Database for more information.

Feature	Description
Oracle Application Express Release 19.2	If your database instance is an Always Free Autonomous Database or is a database with Oracle Database release 19c, then your database has Oracle Application Express Release 19.2. See Creating Applications with Oracle Application Express in Autonomous Database for more information.
Database Resident Connection Pool (DRCP)	Using DRCP provides you with access to a connection pool in your database that enables a significant reduction in key database resources required to support many client connections. See Use Database Resident Connection Pooling with Autonomous Database for more information.

January 2020

Feature	Description
Clone from a backup	Clone from a backup lets you create a clone by selecting a backup from a list of backups, or with a timestamp for a point-in-time to clone. See Clone Autonomous Database from a Backup for more information.
Check and set `MAX_STRING_SIZE` value	By default the database uses extended data types and the value of `MAX_STRING_SIZE` is set to the value `EXTENDED`. To support migration from older Oracle Databases or applications you can set `MAX_STRING_SIZE` to the value `STANDARD`. See Checking and Setting MAX_STRING_SIZE for more information.
Number of concurrent statements increased depending on the service you use	The maximum number of concurrent statements is increased, depending on the connection service you are using. See Predefined Database Service Names for Autonomous Database for more information.
Documentation addition showing available Oracle Database versions	The documentation shows the Oracle Database version available with Autonomous Database by region. See Oracle Database Version Availability by Region for more information.
`DBMS_CLOUD` REST API functions	The `DBMS_CLOUD` REST API functions allow you to make HTTP requests using `DBMS_CLOUD.SEND_REQUEST`. These functions provide a generic API that lets you call REST APIs from supported cloud services. See DBMS_CLOUD REST APIs for more information.
Documentation addition to describe sending mail with `UTL_SMTP`	The documentation is updated to provide the steps and sample code for sending email using `UTL_SMTP`. See Sending Mail with Email Delivery on Autonomous Database for more information.
Use ACLs to control access for Autonomous Database tools	You can now use Virtual Cloud Network, Virtual Cloud Network (OCID), IP address, or CIDR block ACLs to control access to Oracle APEX (APEX), RESTful services, and SQL Developer Web. See Use an Access Control List with Autonomous Database for more information.

2019 What's New

Announcements for the noteworthy changes made to Oracle Autonomous Database Serverless Deployment in the year 2019.

December 2019

Feature	Description
Support for `UTL_HTTP` and `UTL_SMTP` PL/SQL Packages with restrictions	Autonomous Database supports the Oracle Database PL/SQL packages `UTL_HTTP`, `UTL_SMTP`, and `DBMS_NETWORK_ACL_ADMIN` with restrictions. See Restrictions for Database PL/SQL Packages for more information.
Microsoft Active Directory Users	You can configure Autonomous Database to authenticate and authorize Microsoft Active Directory users. This allows Active Directory users to access a database using their Active Directory credentials. See Use Microsoft Active Directory with Autonomous Database for more information.
Validate external partitioned tables and hybrid partitioned tables	You can validate individual partitions of an external partitioned table or a hybrid partitioned table with the procedures `DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE` and `DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE`. See Validate External Partitioned Data and Validate Hybrid Partitioned Data for more information.
Use ACL IP address or CIDR block settings to control access for Autonomous Database tools	You can now use IP address and CIDR based ACLs to control access to Oracle APEX (APEX), RESTful services, and SQL Developer Web. See Use an Access Control List with Autonomous Database for more information.
Use Oracle Data Safe	Oracle Data Safe provides features that help you protect sensitive and regulated data in your database. See Safeguard Your Data with Data Safe on Autonomous Database for more information.

November 2019

Feature	Description
Use ACLs to specify Oracle Cloud Infrastructure VCNs	Use ACLs to specify Oracle Cloud Infrastructure VCNs that can connect to your Autonomous Database.

Feature	Description
SQL Developer Web Data Loading	In SQL Developer Web in the Worksheet page, you can upload data from local files into an existing table.
Access APEX from Oracle Cloud Infrastructure Console	The Tools Tab provides access to APEX from Oracle Cloud Infrastructure Console. See Access Oracle Application Express Administration Services for more information.
Access SQL Developer Web from Oracle Cloud Infrastructure Console	The Tools Tab provides access to SQL Developer Web from Oracle Cloud Infrastructure Console. See Access SQL Developer Web as ADMIN for more information.
Access Oracle Machine Learning from Oracle Cloud Infrastructure Console	The Tools Tab provides access to Oracle Machine Learning User Administration from Oracle Cloud Infrastructure Console.
View Maintenance Schedule from Oracle Cloud Infrastructure Console	The Autonomous Database Information tab shows the schedule for upcoming maintenance. See About Autonomous Database Maintenance Windows for more information.
New fields with `LIST_FILES` and `LIST_OBJECTS`	The functions `DBMS_CLOUD.LIST_FILES` and `DBMS_CLOUD.LIST_OBJECTS` produce additional metadata for files and objects. See LIST_FILES Function and LIST_OBJECTS Function for more information.

October 2019

Feature	Description
Preview Version for Autonomous Database	Oracle periodically provides a preview version of Autonomous Database that allows you to test your applications and to become familiar with features in the next release of Autonomous Database. See Preview Versions for Autonomous Database for more information.
Download database specific instance wallets or regional wallets	You can download database specific instance wallets or regional wallets. See Download Client Credentials (Wallets) for more information.
Rotate database specific instance wallets or regional wallets	You can rotate database specific instance wallets or regional wallets. See Rotate Wallets for Autonomous Database for more information.
Create partitioned external tables with `DBMS_CLOUD`	You can create and validate external partitioned tables with the `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` and `DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE` statements. See Query External Partitioned Data for more information.
Numeric Formats	With the `numberformat` and `numericcharacters` format options Autonomous Database supports formats to interpret numeric strings correctly. See DBMS_CLOUD Package Format Options for more information.

September 2019

Feature	Description
Always Free Autonomous Database	You have the option to create a limited number of Always Free Autonomous Databases that do not consume cloud credits. Always Free databases can be created in Oracle Cloud Infrastructure accounts that are in a trial period, have paying status, or are always free. See Always Free Autonomous Database for more information.
Autonomous Database Metrics	You can monitor the health, capacity, and performance of your databases with metrics, alarms, and, notifications. You can use Oracle Cloud Infrastructure console or Monitoring APIs to view metrics. See Monitor Performance with Autonomous Database Metrics for more information.
Object Store Using Public URL	If your source files reside on an Object Store that provides public URLs, you can use public URLs with `DBMS_CLOUD` procedures. Public means the Object Storage service supports anonymous, unauthenticated access to the Object Store files. See URI Format Using Public URL for more information.
Work Requests	Work Requests are available in the Oracle Cloud Infrastructure console. Work Requests let you track progress of database lifecycle management operations such as creating, terminating, backing up (manual), restoring, scaling, and cloning an Autonomous Database. Work requests allow you to track the progress and steps completed in a database operation. See About Work Requests for more information.

August 2019

Feature	Description
Create Directory and Drop Directory commands	The `data_pump_dir` directory is available in a database. You can use `CREATE DIRECTORY` to create additional directories. To drop a directory, you can use the `DROP DIRECTORY` command. See Creating and Managing Directories for more information.

July 2019

Feature	Description
Performance Hub	Oracle Cloud Infrastructure console includes Performance Hub for Autonomous Database. You can view real-time and historical performance data from the Performance Hub. See Monitor Autonomous Database with Performance Hub for more information.
Oracle Cloud Infrastructure Native Object Store Authentication	Autonomous Database supports native authentication with Oracle Cloud Infrastructure Object Store. Using native authentication you can use key based authentication to access the Object Store (instead of using a username and password). See CREATE_CREDENTIAL Procedure for more information.

Feature	Description
Move to a different Compartment	You can move a database to a different Oracle Cloud Infrastructure compartment. See Move an Autonomous Database to a Different Compartment for more information.

June 2019

Feature	Description
Application Express (APEX)	Oracle APEX (APEX) is a low-code development platform that enables you to build scalable, secure enterprise applications with world-class features that can be deployed anywhere. Each Autonomous Database instance includes a dedicated instance of Oracle APEX; you can use this instance to create multiple workspaces. A workspace is a shared work area where you can build applications. See About Oracle Application Express for more information.
SQL Developer Web	Oracle SQL Developer Web provides a provides a development environment and a data modeler interface for Autonomous Database. See About SQL Developer Web for more information.
Oracle REST Data Services (ORDS)	You can develop and deploy RESTful Services with native Oracle REST Data Services (ORDS) support on a database See Developing RESTful Services in Autonomous Database for more information.
Auto Scaling	Enabling auto scaling allows a database to use up to three times more CPU and IO resources than the currently specified CPU Core Count. When auto scaling is enabled, if your workload requires additional CPU and IO resources the database automatically uses the resources without any manual intervention required.
Azure object store with Data Pump	Autonomous Database now supports Microsoft Azure cloud storage with Oracle Data Pump. See Import Data Using Oracle Data Pump on Autonomous Database for more information.

May 2019

Feature	Description
Support for creating database links	You can create database links with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to access objects on another database. See Use Database Links with Autonomous Database for more information.
Oracle Spatial and Graph with limitations	
Oracle Text with limitations	See Restrictions for Oracle Text for more information.
Oracle Cloud Infrastructure Native URI Format Supported with Oracle Data Pump	For importing data from Data Pump files using `impdp`, if your source files reside on the Oracle Cloud Infrastructure Object Storage you can use the Oracle Cloud Infrastructure native URIs in addition to the Swift URIs. See DBMS_CLOUD Package File URI Formats for more information.

Feature	Description
Oracle Cloud Infrastructure Pre-Authenticated URI Format Supported with `DBMS_CLOUD`	If your source files reside on the Oracle Cloud Infrastructure Object Storage you can use the Oracle Cloud Infrastructure Pre-Authenticated URIs. When you create a pre-authenticated request, a unique URL is generated. You can use a pre-authenticated URL in any `DBMS_CLOUD` procedure that takes a URL to access files in the Oracle Cloud Infrastructure object store, without the need to create a credential. See DBMS_CLOUD Package File URI Formats for more information.
Oracle XML DB with limitations	Oracle XML DB is a high-performance, native XML storage and retrieval technology that is delivered as a part of Oracle Database. Limitations apply to Oracle XML DB with Autonomous Database database. See Restrictions for Oracle XML DB for more information.
Oracle Management Cloud support	Oracle Management Cloud allows you to monitor your Autonomous Database database availability and performance. See *Using Oracle Database Management for Autonomous Databases* for more information.
Modify CPU/IO Shares from Service Console	Autonomous Database comes with predefined CPU/IO shares assigned to different consumer groups. You can modify predefined CPU/IO shares if your workload requires different CPU/IO resource allocations.You can change CPU/IO shares from the Service Console or using a PL/SQL package. See Manage CPU/IO Shares on Autonomous Database for more information.

April 2019

Feature	Description
Support for Network Access Control Lists	Autonomous Database now supports setting a network Access Control List (ACL) to restrict access to a specific Autonomous Database database. When you set a network ACL the database only accepts connections from addresses specified on the ACL and rejects all other client connections. See Configuring Network Access with Access Control Rules (ACLs) for more information.
Support for Updating License Type	You can now update your license type from the Oracle Cloud Infrastructure console **Actions** list. See Update License Type on Autonomous Database for more information.
Access Avro files in Object Stores	Autonomous Database allows you to directly query and load data stored as Apache Avro format files (Apache Parquet format files are also supported). You can create external tables for Parquet or Avro format data files. See DBMS_CLOUD Package Format Options for Parquet and Avro and CREATE_EXTERNAL_TABLE Procedure for Parquet or Avro Files for more information.

Feature	Description
Enhanced newline handling for file record delimiter with `DBMS_CLOUD`	Now, when reading a file, by default `DBMS_CLOUD` tries to automatically find the correct newline character to use as the record delimiter, either for Windows, newline character "`\r\n`", or for UNIX/Linux newline character "`\n`". If `DBMS_CLOUD` finds one of these it sets the record delimiter for the file. You can specify the record delimiter explicitly if you want to override the default behavior. See DBMS_CLOUD Package Format Options for more information.
Modify CPU/IO Shares	Autonomous Database comes with predefined CPU/IO shares assigned to different consumer groups. You can modify these predefined CPU/IO shares if your workload requires different CPU/IO resource allocations. See Manage CPU/IO Shares on Autonomous Database for more information.

March 2019

Feature	Description
Clone Your Database	Autonomous Database provides cloning where you can choose to clone either the full database or only the database metadata. See Cloning a Database with Autonomous Database for more information.
Oracle Cloud Infrastructure Native URI Format Supported with `DBMS_CLOUD`	If your source files reside on the Oracle Cloud Infrastructure Object Storage you can use the Oracle Cloud Infrastructure native URIs in addition to the Swift URIs. See DBMS_CLOUD Package File URI Formats for more information.
UI/API Unification and Workload Type field	Oracle Cloud Infrastructure Console and the Oracle Cloud Infrastructure APIs for Autonomous Data Warehouse and Autonomous Transaction Processing are converged to a single, unified framework. These changes allow you to more easily manage both types of Autonomous Databases. There is a new Workload Type field on the console showing values of either “Transaction Processing” or “Data Warehouse”, depending on the type of database you are viewing. You can also select a Workload Type when you provision an Autonomous Database. See Workload Types for more information.
Service Gateway	You can now set up a Service Gateway with Autonomous Database. A service gateway allows connectivity to Autonomous Database from private IP addresses in private subnets without requiring an Internet Gateway in your VCN. See Access Autonomous Database with Service Gateway and Access to Oracle Services: Service Gateway for more information.
Documentation changes	The procedure `DBMS_CLOUD.DELETE_ALL_OPERATIONS` is now documented. See DELETE_ALL_OPERATIONS Procedure for more information.

February 2019

Feature	Description
Application Continuity	You can now enable and disable Application Continuity. Application Continuity masks outages from end users and applications by recovering the in-flight work for impacted database sessions following outages. Application Continuity performs this recovery beneath the application so that the outage appears to the application as a slightly delayed execution. See Using Application Continuity on Autonomous Database for more information.
Documentation changes	Autonomous Database documentation includes a new chapter that describes moving data to other Oracle databases. See Moving Data from Autonomous Database to Other Oracle Databases for more information.

January 2019

Feature	Description
Access Parquet files in Object Stores	Autonomous Database allows you to directly query and load data stored as parquet files in object stores. You can also create external parquet format tables in object stores. See COPY_DATA Procedure for Parquet Files and CREATE_EXTERNAL_TABLE Procedure for Parquet Files for more information.

2018 What's New

Announcements for the noteworthy changes made to Oracle Autonomous Database Serverless Deployment in the year 2018.

- October 2018
- September 2018
- August 2018
- July 2018
- June 2018
- May 2018
- April 2018

October 2018

Feature	Description
Oracle Cloud Infrastructure Console changes	The Autonomous Database console has a new layout and provides new buttons that improve the usability of Autonomous Database. These changes include a new **DB Connection** button that makes it easier to download client credentials.

September 2018

Feature	Description
Table compression methods	In addition to Hybrid Columnar Compression all table compression types are now available in Autonomous Database. For more information, see Managing DML Performance and Compression.
Idle timeout changes	The idle timeout of 60 minutes is now lifted. Idle sessions that do not hold resources required by other sessions will not be terminated after 60 minutes now. For more information, see Manage Concurrency and Priorities on Autonomous Database.

August 2018

Feature	Description
The Oracle Cloud Infrastructure page has a new option Autonomous Transaction Processing	The sign in for Oracle Cloud Infrastructure now lists the new product, Oracle Autonomous Transaction Processing, in addition to Oracle Autonomous Database.

July 2018

Feature	Description
SQL Developer 18.2.0 and later without Keystore Password field for connections	When you connect to Autonomous Database with SQL Developer 18.2.0 or later, you do not need to supply a keystore password. The keystore password was required in previous SQL Developer versions.

June 2018

Feature	Description
New management interfaces	Autonomous Database is now provisioned and managed using the native Oracle Cloud Infrastructure. This provides a more intuitive user interface to make managing your Autonomous Database instances easier with additional capabilities including sorting and filtering. For more information, see Oracle Help Center.
Better authorization management	With OCI Identity and Access Management (IAM) you can better organize and isolate your Autonomous Database instances using Compartments and control what type of access a user or group of users have.
Built-in auditing	The OCI Audit service records use of Autonomous Database application programming interface (API) endpoints as log events in JSON format. You can view, copy, and analyze audit events with standard log analysis tools or using the Audit service Console, the Audit APIs, or the Java SDK.
Availability of the Phoenix region	Autonomous Database is now available in the Phoenix region in addition to Ashburn and Frankfurt regions.
User Assistance changes	The documentation, videos, and examples are updated on the Oracle Help Center to include the procedures to create and control Autonomous Database instances with the Oracle Cloud Infrastructure Console. A new Related Resources page shows related resources including the Autonomous Data Warehouse Forum .
Idle timeouts in database services	The idle timeouts for three database services, high, medium, and low, have now been relaxed. The previous idle timeout setting of 5 minutes will now apply to only idle sessions that hold resources needed by other active users. See Manage Concurrency and Priorities on Autonomous Database.

May 2018

Feature	Description
Oracle Cloud Infrastructure Object Storage Credentials	The name Swift Password is now Auth token. Use of Swift password in the documentation is now replaced with Auth token.

April 2018

Feature	Description
Microsoft Azure Blob Storage integration	Autonomous Database now supports Microsoft Azure Blob Storage for data loading and querying external data. You can load data from or run queries on files residing on Azure Blob Storage. See Loading Data from Files in the Cloud.

Feature	Description
Manage files on the local file system	You can now list files residing on the local file system on Autonomous Database, see LIST_FILES Function. You can also remove files from the local file system. See DELETE_FILE Procedure.

Sample Star Schema Benchmark (SSB) Queries and Analytic Views

The SSB schema contains the tables: lineorder, customer, supplier, part, and dwdate. The following is a list of sample queries and analytic views you can use against the SSB schema. Note that you need to prefix the table names with the schema name SSB in your queries.

> **Note:**
>
> Both SH and SSB are provided as schema-only users, so you cannot unlock or drop those users or set a password. And the storage of the sample data sets does not count towards your database storage.

Star Schema Benchmark Queries

Star Schema Benchmark Queries

```
select sum(lo_extendedprice*lo_discount) as revenue
from ssb.lineorder, ssb.dwdate
where lo_orderdate = d_datekey
and d_yearmonthnum = 199401
and lo_discount between 4 and 6
and lo_quantity between 26 and 35;

select sum(lo_extendedprice*lo_discount) as revenue
from ssb.lineorder, ssb.dwdate
where lo_orderdate = d_datekey
and d_year = 1993
and lo_discount between 1 and 3
and lo_quantity < 25;

select sum(lo_extendedprice*lo_discount) as revenue
from ssb.lineorder, ssb.dwdate
where lo_orderdate = d_datekey
and d_yearmonthnum = 199401
and lo_discount between 4 and 6
```

```
and lo_quantity between 26 and 35;

select sum(lo_extendedprice*lo_discount) as revenue
from ssb.lineorder, ssb.dwdate
where lo_orderdate = d_datekey
and d_weeknuminyear = 6
and d_year = 1994
and lo_discount between 5 and 7
and lo_quantity between 26 and 35;

select sum(lo_revenue), d_year, p_brand1
from ssb.lineorder, ssb.dwdate, ssb.part, ssb.supplier
where lo_orderdate = d_datekey
and lo_partkey = p_partkey
and lo_suppkey = s_suppkey
and p_category = 'MFGR#12'
and s_region = 'AMERICA'
group by d_year, p_brand1
order by d_year, p_brand1;

select sum(lo_revenue), d_year, p_brand1
from ssb.lineorder, ssb.dwdate, ssb.part, ssb.supplier
where lo_orderdate = d_datekey
and lo_partkey = p_partkey
and lo_suppkey = s_suppkey
and p_brand1 between 'MFGR#2221' and 'MFGR#2228'
and s_region = 'ASIA'
group by d_year, p_brand1
order by d_year, p_brand1;

select sum(lo_revenue), d_year, p_brand1
from ssb.lineorder, ssb.dwdate, ssb.part, ssb.supplier
where lo_orderdate = d_datekey
and lo_partkey = p_partkey
and lo_suppkey = s_suppkey
and p_brand1 = 'MFGR#2221'
and s_region = 'EUROPE'
group by d_year, p_brand1
order by d_year, p_brand1;

select c_nation, s_nation, d_year, sum(lo_revenue) as revenue
from ssb.customer, ssb.lineorder, ssb.supplier, ssb.dwdate
where lo_custkey = c_custkey
and lo_suppkey = s_suppkey
and lo_orderdate = d_datekey
and c_region = 'ASIA' and s_region = 'ASIA'
and d_year >= 1992 and d_year <= 1997
group by c_nation, s_nation, d_year
order by d_year asc, revenue desc;

select c_city, s_city, d_year, sum(lo_revenue) as revenue
from ssb.customer, ssb.lineorder, ssb.supplier, ssb.dwdate
where lo_custkey = c_custkey
and lo_suppkey = s_suppkey
and lo_orderdate = d_datekey
and c_nation = 'UNITED STATES'
```

```
and s_nation = 'UNITED STATES'
and d_year >= 1992 and d_year <= 1997
group by c_city, s_city, d_year
order by d_year asc, revenue desc;

select c_city, s_city, d_year, sum(lo_revenue) as revenue
from ssb.customer, ssb.lineorder, ssb.supplier, ssb.dwdate
where lo_custkey = c_custkey
and lo_suppkey = s_suppkey
and lo_orderdate = d_datekey
and (c_city='UNITED KI1' or c_city='UNITED KI5')
and (s_city='UNITED KI1' or s_city='UNITED KI5')
and d_year >= 1992 and d_year <= 1997
group by c_city, s_city, d_year
order by d_year asc, revenue desc;

select c_city, s_city, d_year, sum(lo_revenue) as revenue
from ssb.customer, ssb.lineorder, ssb.supplier, ssb.dwdate
where lo_custkey = c_custkey
and lo_suppkey = s_suppkey
and lo_orderdate = d_datekey
and (c_city='UNITED KI1' or c_city='UNITED KI5')
and (s_city='UNITED KI1' or s_city='UNITED KI5')
and d_yearmonth = 'Dec1997'
group by c_city, s_city, d_year
order by d_year asc, revenue desc;

select d_year, c_nation, sum(lo_revenue - lo_supplycost) as profit
from ssb.dwdate, ssb.customer, ssb.supplier, ssb.part, ssb.lineorder
where lo_custkey = c_custkey
 and lo_suppkey = s_suppkey
 and lo_partkey = p_partkey
 and lo_orderdate = d_datekey
 and c_region = 'AMERICA'
 and s_region = 'AMERICA'
 and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
group by d_year, c_nation
order by d_year, c_nation;

select d_year, s_nation, p_category, sum(lo_revenue - lo_supplycost) as profit
from ssb.dwdate, ssb.customer, ssb.supplier, ssb.part, ssb.lineorder
where lo_custkey = c_custkey
and lo_suppkey = s_suppkey
and lo_partkey = p_partkey
and lo_orderdate = d_datekey
and c_region = 'AMERICA'
and s_region = 'AMERICA'
and (d_year = 1997 or d_year = 1998)
and (p_mfgr = 'MFGR#1'
or p_mfgr = 'MFGR#2')
group by d_year, s_nation, p_category order by d_year, s_nation, p_category;

select d_year, s_city, p_brand1, sum(lo_revenue - lo_supplycost) as profit
from ssb.dwdate, ssb.customer, ssb.supplier, ssb.part, ssb.lineorder
where lo_custkey = c_custkey
and lo_suppkey = s_suppkey
```

```
and lo_partkey = p_partkey
and lo_orderdate = d_datekey
and c_region = 'AMERICA'
and s_nation = 'UNITED STATES'
and (d_year = 1997 or d_year = 1998)
and p_category = 'MFGR#14'
group by d_year, s_city, p_brand1 order by d_year, s_city, p_brand1;
```

Star Schema Benchmark Analytic Views

SSB Analytic Views

Analytic views make it easy to extend a star schema with a hierarchical business model, aggregation and measure calculation rules, presentation and application-specific metadata that can be used to enhance the content of a data set and to simplify the development of business intelligence applications. The SSB schema includes an analytic view and four hierarchies that use the tables of the star schema. Use the following queries to query the analytic Sample SSB view. Note that the analytic view is in the SSB schema.

```
SELECT
  dwdate_hier.member_name as year,
  part_hier.member_name as part,
  customer_hier.c_region,
  customer_hier.member_name as customer,
  lo_quantity,
  lo_revenue
FROM  ssb.ssb_av
  HIERARCHIES (
    dwdate_hier,
    part_hier,
    customer_hier)
WHERE
  dwdate_hier.d_year = '1998'
  AND dwdate_hier.level_name = 'MONTH'
  AND part_hier.level_name = 'MANUFACTURER'
  AND customer_hier.c_region = 'AMERICA'
  AND customer_hier.level_name = 'NATION'
ORDER BY
  dwdate_hier.hier_order,
  part_hier.hier_order,
  customer_hier.hier_order;

SELECT
  dwdate_hier.member_name as time,
  part_hier.member_name as part,
  customer_hier.member_name as customer,
  supplier_hier.member_name as supplier,
  lo_quantity,
  lo_supplycost
FROM  ssb.ssb_av
  HIERARCHIES (
    dwdate_hier,
    part_hier,
    customer_hier,
```

```
    supplier_hier)
WHERE
  dwdate_hier.d_year = '1998'
  AND dwdate_hier.level_name = 'MONTH'
  AND part_hier.level_name = 'MANUFACTURER'
  AND customer_hier.c_region = 'AMERICA'
  AND customer_hier.c_nation = 'CANADA'
  AND customer_hier.level_name = 'CITY'
  AND supplier_hier.s_region = 'ASIA'
  AND supplier_hier.level_name = 'REGION'
ORDER BY
  dwdate_hier.hier_order,
  part_hier.hier_order,
  customer_hier.hier_order,
  supplier_hier.hier_order;

SELECT
  dwdate_hier.member_name as year,
  part_hier.member_name as part,
  customer_hier.member_name as customer,
  supplier_hier.member_name as supplier,
  lo_quantity,
  lo_revenue,
  lo_supplycost
FROM  ssb.ssb_av
  HIERARCHIES (
    dwdate_hier,
    part_hier,
    customer_hier,
    supplier_hier)
WHERE
  dwdate_hier.d_yearmonth = 'Apr1998'
  AND dwdate_hier.level_name = 'DAY'
  AND part_hier.level_name = 'MANUFACTURER'
  AND customer_hier.c_region = 'AMERICA'
  AND customer_hier.c_nation = 'CANADA'
  AND customer_hier.level_name = 'CITY'
  AND supplier_hier.level_name = 'REGION'
ORDER BY
  dwdate_hier.hier_order,
  part_hier.hier_order,
  customer_hier.hier_order,
  supplier_hier.hier_order;

SELECT
  dwdate_hier.member_name as year,
  part_hier.member_name as part,
  supplier_hier.member_name as supplier,
  lo_quantity,
  lo_extendedprice,
  lo_ordtotalprice,
  lo_revenue,
  lo_supplycost
FROM  ssb.ssb_av
  HIERARCHIES (
    dwdate_hier,
```

```
    part_hier,
    supplier_hier)
WHERE
  dwdate_hier.level_name = 'YEAR'
  AND part_hier.level_name = 'MANUFACTURER'
  AND supplier_hier.level_name = 'SUPPLIER'
  AND supplier_hier.s_suppkey = '23997';

SELECT
  dwdate_hier.member_name as time,
  part_hier.p_container,
  part_hier.member_name as part,
  lo_quantity,
  lo_extendedprice,
  lo_ordtotalprice,
  lo_revenue,
  lo_supplycost
FROM  ssb.ssb_av
  HIERARCHIES (
    dwdate_hier,
    part_hier)
WHERE
  dwdate_hier.member_name = 'June 10, 1998     '
  AND dwdate_hier.level_name = 'DAY'
  AND part_hier.level_name = 'PART'
  AND part_hier.p_size = 32;

SELECT
  dwdate_hier.member_name as time,
  part_hier.member_name as part,
  part_hier.p_name,
  part_hier.p_color,
  lo_quantity,
  lo_revenue,
  lo_supplycost,
  lo_revenue - lo_supplycost as profit
FROM  ssb.ssb_av
  HIERARCHIES (
    dwdate_hier,
    part_hier)
WHERE
  dwdate_hier.d_yearmonth = 'Aug1996'
  AND dwdate_hier.d_dayofweek = 'Friday   '
  AND dwdate_hier.level_name = 'DAY'
  AND part_hier.level_name = 'PART'
  AND part_hier.p_color in ('ivory','coral')
ORDER BY
  dwdate_hier.hier_order,
  part_hier.hier_order;
```

Autonomous Database Supplied Package Reference

This appendix provides information about the packages you use with Autonomous Database. The `DBMS_CLOUD` topic also covers the `DBMS_CLOUD REST APIs`.

- DBMS_CLOUD Package
  The `DBMS_CLOUD` package provides database routines for working with cloud resources.
- DBMS_CLOUD_ADMIN Package
  The `DBMS_CLOUD_ADMIN` package provides administrative routines for configuring a database.
- DBMS_CLOUD_AI Package
  The `DBMS_CLOUD_AI` package facilitates and configures the translation of natural language prompts to SQL statements.
- DBMS_CLOUD_FUNCTION Package
  The `DBMS_CLOUD_FUNCTION` package allows you invoke OCI and AWS Lambda remote functions in your Autonomous Database as SQL functions.
- DBMS_CLOUD_LINK Package
  The `DBMS_CLOUD_LINK` package allows a user to register a table or a view as a data set for read only access with Cloud Links.
- DBMS_CLOUD_LINK_ADMIN Package
- DBMS_CLOUD_MACADM Package
  This section covers the `DBMS_CLOUD_MACADM` subprograms provided with Autonomous Database.
- DBMS_CLOUD_NOTIFICATION Package
  The `DBMS_CLOUD_NOTIFICATION` package allows you to send messages or the output of a SQL query to a provider.
- DBMS_DATA_ACCESS Package
  The `DBMS_DATA_ACCESS` package provides routines to generate and manage Pre-Authenticated Request (PAR) URLs for data sets.
- DBMS_PIPE Package
- DBMS_CLOUD_PIPELINE Package
  The `DBMS_CLOUD_PIPELINE` package allows you to create data pipelines for loading and exporting data in the cloud. This package supports continuous incremental data load of files in object store into the database. `DBMS_CLOUD_PIPELINE` also supports continuous incremental export of table data or query results from the database to object store based on a timestamp column.
- DBMS_CLOUD_REPO Package
  The `DBMS_CLOUD_REPO` package provides for use of and management of cloud hosted code repositories from Oracle Database. Supported cloud code repositories include GitHub, AWS CodeCommit , and Azure Repos.
- DBMS_DCAT Package
  The `DBMS_DCAT` package provides functions and procedures to help Autonomous Database users leverage the data discovery and centralized metadata management system of OCI Data Catalog.
- DBMS_MAX_STRING_SIZE Package
  The `DBMS_MAX_STRING_SIZE` package provides an interface for checking and changing the value of the `DBMS_MAX_STRING_SIZE` initialization parameter.
- DBMS_AUTO_PARTITION Package
  The `DBMS_AUTO_PARTITION` package provides administrative routines for managing automatic partitioning of schemas and tables.
- CS_RESOURCE_MANAGER Package
  The `CS_RESOURCE_MANAGER` package provides an interface to list and update consumer group parameters, and to revert parameters to default values.

- CS_SESSION Package
  The `CS_SESSION` package provides an interface to switch the database service and consumer group of the existing session.
- DBMS_SHARE Package
  The `DBMS_SHARE` subprograms and views are used to share local data with external systems.

DBMS_CLOUD Package

The `DBMS_CLOUD` package provides database routines for working with cloud resources.

- DBMS_CLOUD Endpoint Management
- DBMS_CLOUD Subprograms and REST APIs
  This section covers the `DBMS_CLOUD` subprograms and REST APIs provided with Autonomous Database.
- DBMS_CLOUD URI Formats
  Describes the format of the source file URIs in operations with `DBMS_CLOUD`. The format depends on the object storage service you are using.
- DBMS_CLOUD Package Format Options
  The format argument in `DBMS_CLOUD` specifies the format of source files.
- DBMS_CLOUD Package Format Options for EXPORT_DATA
- DBMS_CLOUD Avro, ORC, and Parquet Support
  This section covers the `DBMS_CLOUD` Avro, ORC, and Parquet support provided with Autonomous Database.
- DBMS_CLOUD Exceptions
  The following table describes exceptions for `DBMS_CLOUD`.

DBMS_CLOUD Endpoint Management

Describes the format of pre-configured endpoint `URIs` in operations with `DBMS_CLOUD`.
The `DBMS_CLOUD` package supports a number of object store and REST endpoints that are all pre-configured for the service. This package also allows you to access additional non-preconfigured endpoints after properly enabling such endpoints through network access control lists (**ACLs**).

The authentication for pre-configured endpoints is automatically understood and derived in `DBMS_CLOUD`. In order to know the proper authentication type for additional, customer-managed endpoints, `DBMS_CLOUD` supports `URI` schemes to indicate the authentication type for the URI endpoint. The `URI` endpoint must support HTTPS on port 443 for secure HTTP requests.

- For more information on pre-configured URIs, see DBMS_CLOUD URI Formats.
- For more information on customer-managed URIs, see Additional Customer-Managed URI Formats

DBMS_CLOUD Subprograms and REST APIs

This section covers the `DBMS_CLOUD` subprograms and REST APIs provided with Autonomous Database.

> **Note:**
>
> To run `DBMS_CLOUD` subprograms with a user other than ADMIN you need to grant `EXECUTE` privileges to that user. For example, run the following command as ADMIN to grant privileges to `adb_user`:
>
> ```
> GRANT EXECUTE ON DBMS_CLOUD TO adb_user;
> ```

The `DBMS_CLOUD` package is made up of the following:

- DBMS_CLOUD for Access Management
  The subprograms for credential management within the DBMS_CLOUD package, including creating, deleting, and updating credentials.
- DBMS_CLOUD for Objects and Files
  The subprograms for object and file management within the DBMS_CLOUD package.
- DBMS_CLOUD for Bulk File Management
  The subprograms for bulk file operations within the DBMS_CLOUD package.
- DBMS_CLOUD REST APIs
  This section covers the `DBMS_CLOUD` REST APIs provided with Autonomous Database.

DBMS_CLOUD for Access Management

The subprograms for credential management within the DBMS_CLOUD package, including creating, deleting, and updating credentials.

Subprogram	Description
CREATE_CREDENTIAL Procedure	This procedure stores cloud service credentials in Autonomous Database.
DROP_CREDENTIAL Procedure	This procedure removes an existing credential from Autonomous Database.
REFRESH_VAULT_CREDENTIAL Procedure	This procedure immediately refreshes the vault secret of a vault secret credential to get the latest version of the vault secret for the specified `credential_name` in Autonomous Database.
UPDATE_CREDENTIAL Procedure	This procedure updates cloud service credential attributes in Autonomous Database.

- CREATE_CREDENTIAL Procedure
  This procedure stores cloud service credentials in Autonomous Database.
- DROP_CREDENTIAL Procedure
  This procedure removes an existing credential from Autonomous Database.
- REFRESH_VAULT_CREDENTIAL Procedure
  This procedure refreshes the vault secret of a vault secret credential.
- UPDATE_CREDENTIAL Procedure
  This procedure updates an attribute with a new value for a specified `credential_name`.

CREATE_CREDENTIAL Procedure

This procedure stores cloud service credentials in Autonomous Database.

Use stored cloud service credentials to access the cloud service for data loading, for querying external data residing in the cloud, or for other cases when you use `DBMS_CLOUD` procedures with a `credential_name` parameter. This procedure is overloaded:

- Use the Oracle Cloud Infrastructure-related parameters, including: `user_ocid`, `tenancy_ocid`, `private_key`, and `fingerprint` only when you are using Oracle Cloud Infrastructure Signing Keys authentication.
- Use the `params` parameter for one of the following:
  - Amazon Resource Names (ARNs) credentials
  - Google Analytics or Google BigQuery credentials
  - Vault secret credentials for use with a supported vault:
    - Oracle Cloud Infrastructure Vault
    - Azure Key Vault
    - AWS Secrets Manager
    - GCP Secret Manager

Syntax

```
DBMS_CLOUD.CREATE_CREDENTIAL (
      credential_name   IN VARCHAR2,
      username          IN VARCHAR2,
      password          IN VARCHAR2 DEFAULT NULL);


DBMS_CLOUD.CREATE_CREDENTIAL (
      credential_name IN VARCHAR2,
      user_ocid       IN VARCHAR2,
      tenancy_ocid    IN VARCHAR2,
      private_key     IN VARCHAR2,
      fingerprint     IN VARCHAR2);


DBMS_CLOUD.CREATE_CREDENTIAL (
      credential_name  IN VARCHAR2,
      params           IN CLOB DEFAULT);
```

Parameters

Parameter	Description
`credential_name`	The name of the credential to be stored. The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
`username`	The `username` and `password` arguments together specify your cloud service credentials. See the usage notes for what to specify for the `username` and `password` for different cloud services.

Parameter	Description
password	The username and password arguments together specify your cloud service credentials.
user_ocid	Specifies the user's OCID. See Where to Get the Tenancy's OCID and User's OCID for details on obtaining the User's OCID.
tenancy_ocid	Specifies the tenancy's OCID. See Where to Get the Tenancy's OCID and User's OCID for details on obtaining the Tenancy's OCID.
private_key	Specifies the generated private key. Private keys generated with a passphrase are not supported. You need to generate the private key without a passphrase. See How to Generate an API Signing Key for details on generating a key pair in PEM format.
fingerprint	Specifies a fingerprint. After a generated public key is uploaded to the user's account the fingerprint is displayed in the console. Use the displayed fingerprint for this argument. See How to Get the Key's Fingerprint and How to Generate an API Signing Key for more details.
params	Specifies credential parameters for one of the following: • Amazon Resource Names (ARNs) credentials • Google Analytics or Google BigQuery credentials • Vault secret credentials for use with username/password type credentials where you store the password in a supported vault: – Oracle Cloud Infrastructure Vault – Azure Key Vault – AWS Secrets Manager – GCP Secret Manager To create a vault secret credential you must have EXECUTE privilege on the DBMS_CLOUD package.

Usage Notes

- This operation stores the credentials in the database in an encrypted format.
- You can see the credentials in your schema by querying the user_credentials table.
- The ADMIN user can see all the credentials by querying the dba_credentials table.
- You only need to create credentials once unless your cloud service credentials change. Once you store the credentials you can then use the same credential name for DBMS_CLOUD procedures that require a credential_name parameter.
- This procedure is overloaded. If you provide one of the key based authentication attributes, user_ocid, tenancy_ocid, private_key, or fingerprint, the call is assumed to be an Oracle Cloud Infrastructure Signing Key based credential.
- You can list credentials from the view ALL_CREDENTIALS. For example, run the following command to list credentials:

```
SELECT credential_name, username, comments FROM all_credentials;
```

Oracle Cloud Infrastructure Credentials (Auth Tokens)

For Oracle Cloud Infrastructure the username is your Oracle Cloud Infrastructure user name. The password is your Oracle Cloud Infrastructure auth token. See Working with Auth Tokens.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adb_user@example.com',
    password => 'password' );
END;
/
```

Use Auth Token based credentials when you are authenticating calls to OCI Object Storage. For calls to any other type of Oracle Cloud Infrastructure cloud service, use Oracle Cloud Infrastructure Signing Key Based Credentials.

Oracle Cloud Infrastructure Signing Key Based Credentials

Use the Oracle Cloud Infrastructure signing key related parameters, including: `user_ocid`, `tenancy_ocid`, `private_key`, and `fingerprint` with Oracle Cloud Infrastructure Signing Keys authentication.

For example:

```
BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL (
       credential_name => 'OCI_KEY_CRED',
       user_ocid       =>
'ocid1.user.oc1..aaaaaaaauq54mi7zdyfhw33ozkwuontjceel7fok5nq3bf2vwetkpqsoa',
       tenancy_ocid    =>
'ocid1.tenancy.oc1..aabbbbbbaafcue47pqmrf4vigneebgbcmmoy5r7xvoypicjqqge32ewnrc
yx2a',
       private_key     =>
'MIIEogIBAAKCAQEAtUnxbmrekwgVac6FdWeRzoXvIpA9+0r1.....wtnNpESQQQ0QLGPD8NM//
JEBg=',
       fingerprint     => 'f2:db:f9:18:a4:aa:fc:94:f4:f6:6c:39:96:16:aa:27');
END;
/
```

Private keys generated with a passphrase are not supported. You need to generate the private key without a passphrase. See How to Generate an API Signing Key for more information.

Oracle Cloud Infrastructure Object Storage Classic Credentials

If your source files reside in Oracle Cloud Infrastructure Object Storage Classic, the `username` is your Oracle Cloud Infrastructure Classic user name and the `password` is your Oracle Cloud Infrastructure Classic password.

Amazon Web Services (AWS) Credentials

If your source files reside in Amazon S3 or you are calling an AWS API, the `username` is your AWS access key ID and the `password` is your AWS secret access key. See AWS Identity and Access Management.

Microsoft Azure Credentials

If your source files reside in Azure Blob Storage or you are calling an Azure API, the `username` is your Azure storage account name and the `password` is an Azure storage account access key. See About Azure storage accounts.

Amazon S3-Compatible Credentials

Service	Credentials Information
Oracle Cloud Infrastructure (Customer Secret Keys)	If your source files reside in Oracle Cloud Infrastructure, then you need to use Customer Secret Keys with S3-compatible URLs. See Working with Customer Secret Keys for more information.
Google Cloud Storage	If your source files reside in Google Cloud Storage or you are calling Google Cloud Storage APIs, then you need to set a default Google project and obtain an HMAC key to create credentials to supply with Google Cloud Storage S3-compatible URLs. Use the HMAC key id as the username, and the HMAC secret as the password. See Projects and HMAC Keys for more information.
Wasabi Hot Cloud Storage	If your source files reside in Wasabi Hot Cloud Storage or you are calling Wasabi Hot Cloud Storage APIs, then you need Access Keys to create credentials to supply with S3-compatible URLs. Use the Wasabi Hot Cloud Storage Access Key as the username, and the Wasabi Hot Cloud Storage Secret Key as the password. See Creating a Wasabi API Access Key Set for more information.

AWS Amazon Resource Names (ARN) Credentials

If your source files reside in Amazon S3 or you are calling an AWS API, use `params` to specify the parameters for the Amazon Resource Names (ARN).

Parameter	Value
`aws_role_arn`	Specifies the Amazon Resource Name (ARN) that identifies the AWS role. If this parameter is not supplied when creating the credential, `ORA-20041` is raised.
`external_id_type`	Optionally set the `external_id_type` to use the Autonomous Database compartment OCID, database OCID, or tenancy OCID by supplying one of: `compartment_ocid`, `database_ocid`, or `tenant_ocid`. If this parameter is not given when creating the credential, the default value is `database_ocid`.

For example:

```
BEGIN DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name      => 'MY_CRED',
        params               => JSON_OBJECT(
               'aws_role_arn'      value 'arn:aws:iam::123456:role/
AWS_ROLE_ARN',
               'external_id_type'  value 'database_ocid'));
END;
/
```

GitHub Personal Access Token

If your source files reside in a GitHub repository or you are calling a GitHub API, the `username` is your GitHub email and the `password` is your GitHub personal access token. See Creating a personal access token for more information.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'MY_GITHUB_CRED',
    username => 'user@example.com',
    password => 'your_personal_access_token' );
END;
/
```

Google Analytics or Google BigQuery Credentials

If you are accessing Google Analytics or Google BigQuery, use the `params` parameter to specify the Google OAuth 2.0 credential parameters.

Parameter	Value
`gcp_oauth2`	Specifies OAuth 2.0 access for Google Analytics or Google BigQuery with a JSON object that includes the following parameters and their values: • `client_id`: See the Google API Console to obtain the client ID. • `client_secret`: See the Google API Console to obtain the client secret. • `refresh_token`: A refresh token allows your application to obtain new access tokens. See Using OAuth 2.0 to Access Google APIs for more information on Google OAuth credentials.

For example:

```
BEGIN DBMS_CLOUD.CREATE_CREDENTIAL(
      credential_name => 'GOOGLE_BIGQUERY_CRED',
      params => JSON_OBJECT('gcp_oauth2' value
                                  JSON_OBJECT(
                                       'client_id'     value 'client_id',
                                       'client_secret' value 'client_secret',
                                       'refresh_token' value
'refresh_token' )));
END;
/
```

Vault Secret Credentials with Oracle Cloud Infrastructure Vault

To create vault secret credentials with Oracle Cloud Infrastructure Vault, use the `params` parameter to specify the required parameters:

- `username`: Specifies the username of any type of username/password credential such as the username of OCI Swift password. For example, if you have a Swift credential with

username as “scott” and password as “*password*”, provide “scott” as the `username` parameter.

- `secret_id`: Is the vault secret ID. Specify the `secret_id` value as the vault secret OCID. See Overview of Vault for more information.
- `region`: Is an optional parameter that specifies the oracle cloud region identifier. The region, when specified, indicates the location where Oracle Cloud Infrastructure Vault secret is located.

  By default, `CREATE_CREDENTIAL` uses the `region` mapped from the region key in the `secret_id`. An example of a region is `us-ashburn-1`.

  See Regions and Availability Domains for a complete list of regions.

For example:

```
BEGIN DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'OCI_SECRET_CRED',
    params => JSON_OBJECT(
                'username'   value 'scott',
                'region'     value 'us-ashburn-1',
                'secret_id'  value 'ocid1.vaultsecret.co1.ap-
mumbai-1.example..aaaaaaaauq5ok5nq3bf2vwetkpqsoa'));
END;
/
```

Notes for using an Oracle Cloud Infrastructure Vault secret to store vault secrets:

- When you use an Oracle Cloud Infrastructure Vault, on the Autonomous Database instance you must enable principal authentication with `DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL`.
- On Oracle Cloud Infrastructure you must specify a policy for the resource principal to access the secret.

To create a vault secret credential you must have `EXECUTE` privilege on the `DBMS_CLOUD` package.

Vault Secret Credentials with Azure Key Vault

To create Azure Key Vault credentials, use the `params` parameter to specify the required parameters:

- `username`: Specifies the username associated with the key.
- `secret_id`: Specifies the secret name.
- `azure_vault_name`: Specifies the name of the vault where the secret is located.

See Create a key vault for more information.

To create a vault secret credential you must have `EXECUTE` privilege on the `DBMS_CLOUD` package.

Vault Secret Credentials with AWS Secrets Manager

To create vault secret credentials with AWS Secrets Manager, use the `params` parameter to specify the required parameters:

- `username`: Specifies the AWS Secrets Manager access key.

- `secret_id`: Is the AWS Secrets Manager AWS ARN.
- `region`: (Optional) Specifies the AWS service region where the vault and secret are located. An example of the AWS region is "us-east-2". The default `region` is the region specified with the ARN in the `secret_id` parameter.

  See Managing AWS Regions for more information.

  To create a vault secret credential you must have `EXECUTE` privilege on the `DBMS_CLOUD` package.

Vault Secret Credentials with GCP Secret Manager

To create GCP Secret Manager credentials, use the `params` parameter to specify the required parameters:

- `username`: Specifies the username associated with the secret.
- `secret_id`: Is the secret name.
- `gcp_project_id`: Specifies the ID of the project where the secret is located.

See Secret Manager for more information.

To create a vault secret credential you must have `EXECUTE` privilege on the `DBMS_CLOUD` package.

DROP_CREDENTIAL Procedure

This procedure removes an existing credential from Autonomous Database.

Syntax

```
DBMS_CLOUD.DROP_CREDENTIAL (
   credential_name     IN VARCHAR2);
```

Parameters

Parameter	Description
`credential_name`	The name of the credential to be removed.

REFRESH_VAULT_CREDENTIAL Procedure

This procedure refreshes the vault secret of a vault secret credential.

This procedure lets you immediately refresh the vault secret of a vault secret credential to get the latest version of the vault secret for the specified `credential_name`.

Syntax

```
DBMS_CLOUD.REFRESH_VAULT_CREDENTIAL (
    credential_name    IN VARCHAR2);
```

Parameters

Parameter	Description
credential_name	The name of the credential to refresh.

Usage Notes

- The ADMIN user can see all the credentials by querying the dba_credentials table.
- You can list credentials from the view ALL_CREDENTIALS. For example, run the following command to list credentials:

```
SELECT credential_name, username, comments FROM all_credentials;
```

Example

```
BEGIN
  DBMS_CLOUD.REFRESH_VAULT_CREDENTIAL(
     credential_name => 'AZURE_SECRET_CRED');
END;
/
```

UPDATE_CREDENTIAL Procedure

This procedure updates an attribute with a new value for a specified credential_name.

Use stored credentials for data loading, for querying external data residing in the Cloud, or wherever you use DBMS_CLOUD procedures with a credential_name parameter.

Syntax

```
DBMS_CLOUD.UPDATE_CREDENTIAL (
    credential_name   IN VARCHAR2,
    attribute         IN VARCHAR2,
    value             IN VARCHAR2);
```

Parameters

Parameter	Description
credential_name	The name of the credential to be updated.

Parameter	Description
attribute	Name of attribute to update. For a username/password type credential, the valid attribute values are: USERNAME and PASSWORD. For a credential for an Amazon ARN, the valid attribute values are: aws_role_arn and external_id_type. For a credential for Google BigQuery or Google Analytics, the valid attribute values are: client_id, client_secret, and refresh_token. Depending on the vault you are using, for Vault Secret Credentials the valid attribute values are: • Oracle Cloud Infrastructure Vault: secret_id, region • Azure Key Vault: secret_id, azure_vault_name • AWS Secrets Manager: secret_id, region • GCP Secret Manager: secret_id, gcp_project_id See CREATE_CREDENTIAL Procedure for more information.
value	New value for the specified attribute.

Usage Notes

- The username value is case sensitive. It cannot contain double quotes or spaces.
- The ADMIN user can see all the credentials by querying dba_credentials.
- You only need to create credentials once unless your cloud service credentials change. Once you store the credentials you can then use the same credential name for DBMS_CLOUD procedures that require a credential_name parameter.
- You can list credentials from the view ALL_CREDENTIALS. For example, run the following command to list credentials:

```
SELECT credential_name, username, comments FROM all_credentials;
```

Examples

```
BEGIN
  DBMS_CLOUD.UPDATE_CREDENTIAL(
     credential_name => 'OBJ_STORE_CRED',
     attribute => 'PASSWORD',
     value => 'password');
END;
/


BEGIN
  DBMS_CLOUD.UPDATE_CREDENTIAL(
     credential_name => 'ARN_CRED',
     attribute => 'aws_role_arn',
     value => 'NEW_AWS_ARN');
END;
/
```

DBMS_CLOUD for Objects and Files

The subprograms for object and file management within the DBMS_CLOUD package.

Subprogram	Description
COPY_COLLECTION Procedure	This procedure loads data into existing SODA collection either from Cloud Object Storage or from files in a directory.
COPY_DATA Procedure	This procedure loads data into existing Autonomous Database tables either from Cloud Object Storage or from files in a directory.
COPY_DATA Procedure for Avro, ORC, or Parquet Files	This procedure with the `format` parameter `type` set to the value `orc`, `parquet`, or `avro` loads data into existing Autonomous Database tables from ORC, Parquet, or Avro files in the Cloud or from ORC, Parquet, or Avro files in a directory. Similar to text files, the data is copied from the source ORC, Parquet, or Avro file into the preexisting internal table.
COPY_OBJECT Procedure	This procedure copies files from one Cloud Object Storage bucket to another.
CREATE_EXTERNAL_TABLE Procedure	This procedure creates an external table on files in the Cloud or on files in a directory. This allows you to run queries on external data from Autonomous Database.
CREATE_CLOUD_TABLE Procedure	This procedure creates a cloud table where all persistent data is stored in Oracle-Managed Object Storage.
CREATE_EXTERNAL_TABLE Procedure for Apache Iceberg	This procedure creates external tables for Apache Iceberg tables in the supported configurations.
CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files	This procedure with the `format` parameter `type` set to the value `parquet`, `orc`, or `avro`, creates an external table with either Parquet, ORC, or Avro format files in the Cloud or in a directory. This allows you to run queries on external data from Autonomous Database.
CREATE_EXTERNAL_PART_TABLE Procedure	This procedure creates an external partitioned table on files in the Cloud. This allows you to run queries on external data from Autonomous Database.
CREATE_EXTERNAL_TEXT_INDEX Procedure	This procedure creates text index on the object store files.
CREATE_HYBRID_PART_TABLE Procedure	This procedure creates a hybrid partitioned table. This allows you to run queries on hybrid partitioned data from Autonomous Database.
DELETE_ALL_OPERATIONS Procedure	This procedure clears either all data load operations logged in the `user_load_operations` table in your schema or clears all the data load operations of the specified type, as indicated with the `type` parameter.
DELETE_FILE Procedure	This procedure removes the specified file from the specified directory on Autonomous Database
DELETE_OBJECT Procedure	This procedure deletes the specified object on object store.
DROP_EXTERNAL_TEXT_INDEX Procedure	This procedure drops text index on the object store files.

Subprogram	Description
EXPORT_DATA Procedure	This procedure exports data from Autonomous Database to files in the Cloud based on the result of a query. The overloaded form enables you to use the `operation_id` parameter. Depending on the `format` parameter `type` option specified, the procedure exports rows to the Cloud Object store as text with options of CSV, JSON, Parquet, or XML; or using the ORACLE_DATAPUMP access driver to write data to a dump file.
GET_OBJECT Procedure and Function	This procedure is overloaded. The procedure form reads an object from Cloud Object Storage and copies it to Autonomous Database. The function form reads an object from Cloud Object Storage and returns a `BLOB` to Autonomous Database.
LIST_FILES Function	This function lists the files in the specified directory. The results include the file names and additional metadata about the files such as file size in bytes, creation timestamp, and the last modification timestamp.
LIST_OBJECTS Function	This function lists objects in the specified location on object store. The results include the object names and additional metadata about the objects such as size, checksum, creation timestamp, and the last modification timestamp.
MOVE_OBJECT Procedure	This procedure moves an object from one Cloud Object Storage bucket to another one.
PUT_OBJECT Procedure	This procedure is overloaded. In one form the procedure copies a file from Autonomous Database to the Cloud Object Storage. In another form the procedure copies a `BLOB` from Autonomous Database to the Cloud Object Storage.
SYNC_EXTERNAL_PART_TABLE Procedure	This procedure simplifies updating an external partitioned table from files in the Cloud. Run this procedure whenever new partitions are added or when partitions are removed from the Object Store source for the external partitioned table.
VALIDATE_EXTERNAL_TABLE Procedure	This procedure validates the source files for an external table, generates log information, and stores the rows that do not match the format options specified for the external table in a *badfile* table on Autonomous Database.
VALIDATE_EXTERNAL_PART_TABLE Procedure	This procedure validates the source files for an external partitioned table, generates log information, and stores the rows that do not match the format options specified for the external table in a *badfile* table on Autonomous Database.
VALIDATE_HYBRID_PART_TABLE Procedure	This procedure validates the source files for a hybrid partitioned table, generates log information, and stores the rows that do not match the format options specified for the hybrid table in a *badfile* table on Autonomous Database.

- COPY_COLLECTION Procedure
  This procedure loads data into a SODA collection from Cloud Object Storage or from a directory. If the specified SODA collection does not exist, the procedure creates it. The overloaded form enables you to use the `operation_id` parameter.
- COPY_DATA Procedure
  This procedure loads data into existing Autonomous Database tables from files in the Cloud, or from files in a directory. The overloaded form enables you to use the `operation_id` parameter.

- COPY_DATA Procedure for Avro, ORC, or Parquet Files
- COPY_OBJECT Procedure
  This procedure copies an object from one Cloud Object Storage bucket or folder to another.
- CREATE_CLOUD_TABLE Procedure
  This procedure creates a Cloud Table. All Cloud Table data is stored in Oracle managed Object Storage (Cloud Tables only store their metadata in the database).
- CREATE_EXTERNAL_PART_TABLE Procedure
  This procedure creates an external partitioned table on files in the Cloud, or from files in a directory. This allows you to run queries on external data from Autonomous Database.
- CREATE_EXTERNAL_TABLE Procedure
  This procedure creates an external table on files in the Cloud or from files in a directory. This allows you to run queries on external data from Autonomous Database.
- CREATE_EXTERNAL_TABLE Procedure for Apache Iceberg
  This procedure creates external tables for Apache Iceberg tables in the supported configurations.
- CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files
- CREATE_EXTERNAL_TEXT_INDEX Procedure
  This procedure creates a text index on Object Storage files.
- CREATE_HYBRID_PART_TABLE Procedure
  This procedure creates a hybrid partitioned table. This allows you to run queries on hybrid partitioned data from Autonomous Database using database objects and files in the Cloud, or database objects and files in a directory.
- DELETE_ALL_OPERATIONS Procedure
  This procedure clears either all data load operations logged in the `user_load_operations` table in your schema or clears all the data load operations of the specified type, as indicated with the `type` parameter.
- DELETE_FILE Procedure
  This procedure removes the specified file from the specified directory on Autonomous Database.
- DELETE_OBJECT Procedure
  This procedure deletes the specified object on object store.
- DROP_EXTERNAL_TEXT_INDEX Procedure
  This procedure drops text index on the Object Storage files.
- EXPORT_DATA Procedure
- GET_OBJECT Procedure and Function
  This procedure is overloaded. The procedure form reads an object from Cloud Object Storage and copies it to Autonomous Database. The function form reads an object from Cloud Object Storage and returns a `BLOB` to Autonomous Database.
- LIST_FILES Function
  This function lists the files in the specified directory. The results include the file names and additional metadata about the files such as file size in bytes, creation timestamp, and the last modification timestamp.
- LIST_OBJECTS Function
  This function lists objects in the specified location on object store. The results include the object names and additional metadata about the objects such as size, checksum, creation timestamp, and the last modification timestamp.

- MOVE_OBJECT Procedure
  This procedure moves an object from one Cloud Object Storage bucket or folder to another.
- PUT_OBJECT Procedure
  This procedure is overloaded. In one form the procedure copies a file from Autonomous Database to the Cloud Object Storage. In another form the procedure copies a `BLOB` from Autonomous Database to the Cloud Object Storage.
- SYNC_EXTERNAL_PART_TABLE Procedure
  This procedure simplifies updating an external partitioned table from files in the Cloud. Run this procedure whenever new partitions are added or when partitions are removed from the Object Store source for the external partitioned table.
- VALIDATE_EXTERNAL_PART_TABLE Procedure
  This procedure validates the source files for an external partitioned table, generates log information, and stores the rows that do not match the format options specified for the external table in a *badfile* table on Autonomous Database. The overloaded form enables you to use the `operation_id` parameter.
- VALIDATE_EXTERNAL_TABLE Procedure
  This procedure validates the source files for an external table, generates log information, and stores the rows that do not match the format options specified for the external table in a *badfile* table on Autonomous Database. The overloaded form enables you to use the `operation_id` parameter.
- VALIDATE_HYBRID_PART_TABLE Procedure
  This procedure validates the source files for a hybrid partitioned table, generates log information, and stores the rows that do not match the format options specified for the hybrid table in a *badfile* table on Autonomous Database. The overloaded form enables you to use the `operation_id` parameter.

COPY_COLLECTION Procedure

This procedure loads data into a SODA collection from Cloud Object Storage or from a directory. If the specified SODA collection does not exist, the procedure creates it. The overloaded form enables you to use the `operation_id` parameter.

Syntax

```
DBMS_CLOUD.COPY_COLLECTION (
    collection_name   IN VARCHAR2,
    credential_name   IN VARCHAR2 DEFAULT NULL,
    file_uri_list     IN CLOB,
    format            IN CLOB     DEFAULT NULL
);

DBMS_CLOUD.COPY_COLLECTION (
    collection_name   IN VARCHAR2,
    credential_name   IN VARCHAR2 DEFAULT NULL,
    file_uri_list     IN CLOB,
    format            IN CLOB     DEFAULT NULL,
    operation_id      OUT NOCOPY NUMBER
);
```

Parameters

Parameter	Description
`collection_name`	The name of the SODA collection into which data will be loaded. If a collection with this name already exists, the specified data will be loaded, otherwise a new collection is created.
`credential_name`	The name of the credential to access the Cloud Object Storage. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. This parameter is not used when you specify a directory with `file_uri_list`.

Parameter	Description
file_uri_list	This parameter specifies either a comma-delimited list of source file URIs or one or more directories and source files. **Cloud source file URIs** You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the regexuri format parameter is set to TRUE. The characters "*" and "?" are considered wildcard characters when the regexuri parameter is set to FALSE. When the regexuri parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the REGEXP_LIKE function. For example: `file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/)*[a-z]+[1-3]??.csv'` The format of the URIs depends on the Cloud Object Storage service you are using, for details see DBMS_CLOUD URI Formats. See REGEXP_LIKE Condition for more information on REGEXP_LIKE condition. **Directory** You can specify one directory and one or more file names or use a comma separated list of directories and file names. The format to specify a directory is: '*MY_DIR*:*filename.ext*'. By default the directory name *MY_DIR* is a database object and is case-insensitive. The file name is case sensitive. Regular expressions are not supported when specifying the file names in a directory. You can only use wildcards to specify file names in a directory. The character "*" can be used as the wildcard for multiple characters, and the character "?" can be used as the wildcard for a single character. For example: '*MY_DIR*:*" or '*MY_DIR*:test?' To specify multiple directories, use a comma separated list of directories: For example: '*MY_DIR1*:*, *MY_DIR2*:test?' Use double quotes to specify a case-sensitive directory name. For example: '"*my_dir1*":*, "*my_dir2*":Test?' To include a quote character, use two quotes. For example: '*MY_DIR*:''filename.ext'. This specifies the filename starts with a quote ('').
format	The options describing the format of the source files. These options are specified as a JSON string. Supported formats are: characterset, compression, encryption, ignoreblanklines, jsonpath, maxdocsize, recorddelimiter, rejectlimit, type, unpackarrays, keyassignment, and keypath. Apart from the mentioned formats for JSON data, Autonomous Database supports other formats too. For the list of format arguments supported by Autonomous Database, see DBMS_CLOUD Package Format Options.

Parameter	Description
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Example

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
            credential_name => 'OBJ_STORE_CRED',
            username        => 'user_name@oracle.com',
            password        => 'password'
            );

    DBMS_CLOUD.COPY_COLLECTION(
            collection_name => 'myCollection',
            credential_name => 'OBJ_STORE_CRED',
            file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/adbexample/b/json/o/myCollection.json'
            );
END;
/
```

COPY_DATA Procedure

This procedure loads data into existing Autonomous Database tables from files in the Cloud, or from files in a directory. The overloaded form enables you to use the operation_id parameter.

Syntax

```
DBMS_CLOUD.COPY_DATA (
    table_name         IN VARCHAR2,
    credential_name    IN VARCHAR2 DEFAULT NULL,
    file_uri_list      IN CLOB,
    schema_name        IN VARCHAR2,
    field_list         IN CLOB,
    format             IN CLOB);

DBMS_CLOUD.COPY_DATA (
    table_name         IN VARCHAR2,
    credential_name    IN VARCHAR2 DEFAULT NULL,
    file_uri_list      IN CLOB DEFAULT NULL,
    schema_name        IN VARCHAR2 DEFAULT NULL,
    field_list         IN CLOB DEFAULT NULL,
    format             IN CLOB DEFAULT NULL
    operation_id       OUT NOCOPY NUMBER);
```

Parameters

Parameter	Description
table_name	The name of the target table on the database. The target table needs to be created before you run COPY_DATA.

Parameter	Description
credential_name	The name of the credential to access the Cloud Object Storage. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. This parameter is not used when you specify a directory with file_uri_list.
file_uri_list	You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. **Cloud source file URIs** This parameter specifies either a comma-delimited list of source file URIs or one or more directories and source files. Regular expressions can only be used when the regexuri format parameter is set to TRUE. The characters "*" and "?" are considered wildcard characters when the regexuri parameter is set to FALSE. When the regexuri parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the REGEXP_LIKE function. For example: `file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/)*[a-z]+[1-3]??.csv'` The format of the URIs depends on the Cloud Object Storage service you are using, for details see DBMS_CLOUD URI Formats. See REGEXP_LIKE Condition for more information on REGEXP_LIKE condition. **Directory** You can specify one directory and one or more file names or use a comma separated list of directories and file names. The format to specify a directory is: '*MY_DIR*:*filename.ext*'. By default the directory name *MY_DIR* is a database object and is case-insensitive. The file name is case sensitive. Regular expressions are not supported when specifying the file names in a directory. You can only use wildcards to specify file names in a directory. The character "*" can be used as the wildcard for multiple characters, and the character "?" can be used as the wildcard for a single character. For example: '*MY_DIR*:*" or '*MY_DIR*:test?' To specify multiple directories, use a comma separated list of directories: For example: '*MY_DIR1*:*, *MY_DIR2*:test?' Use double quotes to specify a case-sensitive directory name. For example: '"*my_dir1*":*, "*my_dir2*":Test?' To include a quote character, use two quotes. For example: '*MY_DIR*:''filename.ext'. This specifies the filename starts with a quote ('').

Parameter	Description
schema_name	The name of the schema where the target table resides. The default value is NULL meaning the target table is in the same schema as the user running the procedure.
field_list	Identifies the fields in the source files and their data types. The default value is NULL meaning the fields and their data types are determined by the target table definition. This argument's syntax is the same as the field_list clause in regular Oracle external tables. For more information about field_list see *Oracle® Database Utilities*. When the format parameter type option value is json, this parameter is ignored. For an example using field_list, see CREATE_EXTERNAL_TABLE Procedure.
format	The options describing the format of the source, log, and bad files. For the list of the options and how to specify the values see DBMS_CLOUD Package Format Options. For Avro, ORC, or Parquet file format options, see DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet.
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Usage Note

The default record delimiter is detected newline. With detected newline, DBMS_CLOUD tries to automatically find the correct newline character to use as the record delimiter. DBMS_CLOUD first searches for the Windows newline character \r\n. If it finds the Windows newline character, this is used as the record delimiter for all files in the procedure. If a Windows newline character is not found, DBMS_CLOUD searches for the UNIX/Linux newline character \n, and if it finds one it uses \n as the record delimiter for all files in the procedure. If the source files use a combination of different record delimiters, you may encounter an error such as, "KUP-04020: found record longer than buffer size supported". In this case, you need to either modify the source files to use the same record delimiter or only specify the source files that use the same record delimiter.

See DBMS_CLOUD Package Format Options for information on the recorddelmiter format option.

Example

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
            credential_name => 'OBJ_STORE_CRED',
            username        => 'user_name@oracle.com',
            password        => 'password'
            );

    DBMS_CLOUD.COPY_COLLECTION(
            table_name      => 'ORDERS',
            schema_name     => 'TEST_SCHEMA',
            credential_name => 'OBJ_STORE_CRED',
         file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/adbexample/b/json/o/orde[r]s.tbl.1'
            format          =>  json_object('ignoreblanklines' value TRUE,
```

```
                                                    'rejectlimit' value '0',
                                                    'dateformat' value 'yyyy-mm-dd',
                                                    'regexuri' value TRUE)
            );
END;
/
```

COPY_DATA Procedure for Avro, ORC, or Parquet Files

This procedure with the `format` parameter `type` set to the value `avro`, `orc`, or `parquet` loads data into existing Autonomous Database tables from Avro, ORC, or Parquet files in the Cloud or from files in a directory.

Similar to text files, the data is copied from the source Avro, ORC, or Parquet file into the preexisting internal table.

Syntax

```
DBMS_CLOUD.COPY_DATA (
    table_name        IN VARCHAR2,
    credential_name   IN VARCHAR2 DEFAULT NULL,
    file_uri_list     IN CLOB,
    schema_name       IN VARCHAR2 DEFAULT,
    field_list        IN CLOB DEFAULT,
    format            IN CLOB DEFAULT);
```

Parameters

Parameter	Description
`table_name`	The name of the target table on the database. The target table needs to be created before you run `COPY_DATA`.
`credential_name`	The name of the credential to access the Cloud Object Storage. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. This parameter is not used when you specify a directory with `file_uri_list`.

Parameter	Description
file_uri_list	This parameter specifies either a comma-delimited list of source file URIs or one or more directories and source files. **Cloud source file URIs** You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the regexuri format parameter is set to TRUE. The characters "*" and "?" are considered wildcard characters when the regexuri parameter is set to FALSE. When the regexuri parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the REGEXP_LIKE function. For example: `file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/)*[a-z]+[1-3]??.csv'` The format of the URIs depends on the Cloud Object Storage service you are using, for details see DBMS_CLOUD URI Formats. See REGEXP_LIKE Condition for more information on REGEXP_LIKE condition. **Directory** You can specify one directory and one or more file names or use a comma separated list of directories and file names. The format to specify a directory is: '*MY_DIR*:*filename.ext*'. By default the directory name *MY_DIR* is a database object and is case-insensitive. The file name is case sensitive. Regular expressions are not supported when specifying the file names in a directory. You can only use wildcards to specify file names in a directory. The character "*" can be used as the wildcard for multiple characters, and the character "?" can be used as the wildcard for a single character. For example: '*MY_DIR*:*" or '*MY_DIR*:test?' To specify multiple directories, use a comma separated list of directories: For example: '*MY_DIR1*:*, *MY_DIR2*:test?' Use double quotes to specify a case-sensitive directory name. For example: '"*my_dir1*":*, "*my_dir2*":Test?' To include a quote character, use two quotes. For example: '*MY_DIR*:''filename.ext'. This specifies the filename starts with a quote (') .
schema_name	The name of the schema where the target table resides. The default value is NULL meaning the target table is in the same schema as the user running the procedure.

Parameter	Description
field_list	Ignored for Avro, ORC, or Parquet files. The fields in the source match the external table columns by name. Source data types are converted to the external table column data type. For ORC files, see DBMS_CLOUD Package ORC to Oracle Data Type Mapping. For Parquet files, see DBMS_CLOUD Package Parquet to Oracle Data Type Mapping for details on mapping. For Avro files, see DBMS_CLOUD Package Avro to Oracle Data Type Mapping for details on mapping.
format	The options describing the format of the source files. For Avro, ORC, or Parquet files, only two options are supported: see DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet.

Usage Notes

- As with other data files, Avro, ORC, and Parquet data loads generate logs that are viewable in the tables dba_load_operations and user_load_operations. Each load operation adds a record to dba[*user*]_load_operations that indicates the table containing the logs.

  The log table provides summary information about the load.

- For Avro, ORC, or Parquet, when the format parameter type is set to the value avro, orc, or parquet, the BADFILE_TABLE table is always empty.
  - For Parquet files, PRIMARY KEY constraint errors throw an ORA error.
  - If data for a column encounters a conversion error, for example, the target column is not large enough to hold the converted value, the value for the column is set to NULL. This does not produce a rejected record.

COPY_OBJECT Procedure

This procedure copies an object from one Cloud Object Storage bucket or folder to another.

The source and target bucket or folder can be in the same or different Cloud Object store provider.

When the source and target are in distinct Object Stores or have different accounts with the same cloud provider, you can give separate credential names for the source and target locations.

The source credential name is by default also used by the target location when target credential name is not provided.

Syntax

```
DBMS_CLOUD.COPY_OBJECT (
    source_credential_name  IN  VARCHAR2 DEFAULT NULL,
    source_object_uri       IN  VARCHAR2,
    target_object_uri       IN  VARCHAR2,
    target_credential_name  IN  VARCHAR2 DEFAULT NULL
);
```

Parameters

Parameter	Description
source_credential_name	The name of the credential to access the source Cloud Object Storage. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. If you do not supply a source_credential_name value, the credential_name is set to NULL.
source_object_uri	Specifies URI, that point to the source Object Storage bucket or folder location. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD URI Formats for more information.
target_object_uri	Specifies the URI for the target Object Store. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD URI Formats for more information.
target_credential_name	The name of the credential to access the target Cloud Object Storage location. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. If you do not supply a target_credential_name value, the target_object_uri is set to the source_credential_name value.

Example

```
BEGIN
DBMS_CLOUD.COPY_OBJECT (
    source_credential_name => 'OCI_CRED',
    source_object_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/bgfile.csv',
    target_object_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/myfile.csv'
);
END;
/
```

CREATE_CLOUD_TABLE Procedure

This procedure creates a Cloud Table. All Cloud Table data is stored in Oracle managed Object Storage (Cloud Tables only store their metadata in the database).

Syntax

```
DBMS_CLOUD.CREATE_CLOUD_TABLE (
    table_name       IN VARCHAR2,
    column_list      IN CLOB,
    params           IN CLOB);
```

Parameters

Parameter	Description
table_name	The name of the Cloud Table.
column_list	Comma-delimited list of column names and data types for the Cloud Table.

Usage Notes

- DEFAULT attributes: The column_list can include DEFAULT clause, which functions like the DEFAULT clause in an ordinary CREATE TABLE. See CREATE TABLE for information on the behavior of the DEFAULT clause.
- Use DROP TABLE to drop a Cloud Table. Cloud Tables do not support the recycle bin.

  For example:

  ```
  DROP TABLE CLOUD_TAB1;
  ```

- You can grant SELECT, INSERT, and UPDATE privileges for a Cloud Table. No other privileges can be granted to a Cloud Table.

  See Configuring Privilege and Role Authorization for more information.

Examples

```
EXEC DBMS_CLOUD.CREATE_CLOUD_TABLE( 'CLOUD_TAB1', 'I INTEGER, J INTEGER' );


BEGIN
  DBMS_CLOUD.CREATE_CLOUD_TABLE(
   table_name  => 'CLOUD_TABLE_WITH_DEFAULT',
   column_list => 'I INTEGER,
                   A VARCHAR2(32) DEFAULT ''ABC''' );
END;
/
```

CREATE_EXTERNAL_PART_TABLE Procedure

This procedure creates an external partitioned table on files in the Cloud, or from files in a directory. This allows you to run queries on external data from Autonomous Database.

Syntax

```
DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE (
    table_name           IN VARCHAR2,
    credential_name      IN VARCHAR2,
    partitioning_clause  IN CLOB,
    column_list          IN CLOB,
    field_list           IN CLOB DEFAULT,
    format               IN CLOB DEFAULT);


DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE (
```

```
    table_name           IN VARCHAR2,
    credential_name      IN VARCHAR2,
    file_uri_list        IN VARCHAR2,
    column_list          IN CLOB,
    field_list           IN CLOB DEFAULT,
    format               IN CLOB DEFAULT);
```

Parameters

Parameter	Description
`table_name`	The name of the external table.
`credential_name`	The name of the credential to access the Cloud Object Storage. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information.
`partitioning_clause`	Specifies the complete partitioning clause, including the location information for individual partitions. If you use the `partitioning_clause` parameter, the `file_uri_list` parameter is not allowed.
`file_uri_list`	This parameter specifies either a comma-delimited list of source file URIs or one or more directories and source files. **Cloud source file URIs** You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the `regexuri format` parameter is set to `TRUE`. The characters "*" and "?" are considered wildcard characters when the `regexuri` parameter is set to `FALSE`. When the `regexuri` parameter is set to `TRUE` the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the `REGEXP_LIKE` function. This option is only supported with external tables that are created on a file in the Object Storage. For example: `file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/)*[a-z]+[1-3]??.csv'` If you use the parameter `file_uri_list`, the `partitioning_clause` parameter is not allowed. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD URI Formats for more information. See REGEXP_LIKE Condition for more information on `REGEXP_LIKE` condition.

Parameter	Description
column_list	Comma-delimited list of column names and data types for the external table. This parameter has the following requirements, depending on the type of the data files specified with the file_uri_list parameter: • The column_list parameter is required with unstructured files. Using unstructured files, for example with CSV text files, the column_list parameter must specify all the column names and data types inside the data file as well as the partition columns derived from the object name. • The column_list parameter is optional with structured files. For example, with Avro, ORC, or Parquet data files, the column_list is not required. When the column_list is not included, the format parameter partition_columns option must include specifications for both column names (name) and data types (type).
field_list	Identifies the fields in the source files and their data types. The default value is NULL meaning the fields and their data types are determined by the *column_list* parameter. This argument's syntax is the same as the field_list clause in regular Oracle external tables. For more information about field_list see *Oracle® Database Utilities*.

Parameter	Description
format	The format option partition_columns specifies the DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE column names and data types of partition columns when the partition columns are derived from the file path, depending on the type of data file, structured or unstructured: • When the DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE includes the column_list parameter and the data files are unstructured, such as with CSV text files, partition_columns does not include the data type. For example, use a format such as the following for this type of partition_columns specification: '"partition_columns":["state","zipcode"]' The data type is not required because it is specified in the DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE column_list parameter. • When the DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE does not include the column_list parameter and the data files are structured, such as Avro, ORC, or Parquet files, the partition_columns option includes both the column name, name sub-clause, and the data type, type sub-clause. For example, the following shows a partition_columns specification: '"partition_columns":[{"name":"country", "type":"varchar2(10)"}, {"name":"year", "type":"number"}, {"name":"month", "type":"varchar2(10)"}]' If the data files are unstructured and the type sub-clause is specified with partition_columns, the type sub-clause is ignored. For object names that are not based on hive format, the order of the partition_columns specified columns must match the order as they appear in the object name in the file path specified in the file_uri_list parameter. To see all the format parameter options describing the format of the source files, see DBMS_CLOUD Package Format Options.

Usage Notes

- You cannot call this procedure with both partitioning_clause and file_uri_list parameters.
- Specifying the column_list parameter is optional with structured data files, including Avro, Parquet, or ORC data files. If column_list is not specified, the format parameter partition_columns option must include both name and type.
- The column_list parameter is required with unstructured data files, such as CSV text files.
- The procedure DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE supports external partitioned files in the supported cloud object storage services, including:
  - Oracle Cloud Infrastructure Object Storage

  - Azure Blob Storage
  - Amazon S3
  - Amazon S3-Compatible, including: Oracle Cloud Infrastructure Object Storage, Google Cloud Storage, and Wasabi Hot Cloud Storage.
  - GitHub Repository

  See DBMS_CLOUD URI Formats for more information.

- The procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` supports external partitioned files in directories, either in a local file system or in a network file system.
- When you call `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` with the `file_uri_list` parameter, the types for columns specified in the Cloud Object Store file name must be one of the following types:

  ```
  VARCHAR2(n)
  NUMBER(n)
  NUMBER(p,s)
  NUMBER
  DATE
  TIMESTAMP(9)
  ```

- The default record delimiter is `detected newline`. With `detected newline`, `DBMS_CLOUD` tries to automatically find the correct newline character to use as the record delimiter. `DBMS_CLOUD` first searches for the Windows newline character `\r\n`. If it finds the Windows newline character, this is used as the record delimiter for all files in the procedure. If a Windows newline character is not found, `DBMS_CLOUD` searches for the UNIX/Linux newline character `\n`, and if it finds one it uses `\n` as the record delimiter for all files in the procedure. If the source files use a combination of different record delimiters, you may encounter an error such as, "`KUP-04020: found record longer than buffer size supported`". In this case, you need to either modify the source files to use the same record delimiter or only specify the source files that use the same record delimiter.

  See DBMS_CLOUD Package Format Options for information on the `recorddelmiter` format option.
- The external partitioned tables you create with `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from.
  - `file$path`: Specifies the file path text up to the beginning of the object name.
  - `file$name`: Specifies the object name, including all the text that follows the bucket name.

Examples

Example using the `partitioning_clause` parameter:

```
BEGIN
   DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
      table_name =>'PET1',
      credential_name =>'OBJ_STORE_CRED',
      format => json_object('delimiter' value ',', 'recorddelimiter' value
'newline', 'characterset' value 'us7ascii'),
      column_list => 'col1 number, col2 number, col3 number',
      partitioning_clause => 'partition by range (col1)
```

```
                                        (partition p1 values less than (1000) location
                                            ( ''&base_URL//file_11.txt'')
                                         ,
                                         partition p2 values less than (2000) location
                                            ( ''&base_URL/file_21.txt'')
                                         ,
                                         partition p3 values less than (3000)
location
                                            ( ''&base_URL/file_31.txt'')
                                         )'

     );
   END;
/


BEGIN
    DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
       table_name          => 'PET',
       format              => json_object('delimiter'value ','),
       column_list         => 'name varchar2(20), gender varchar2(10), salary
number',
       partitioning_clause => 'partition by range (salary)
              (   -- Use test1.csv in the DEFAULT DIRECTORY DATA_PUMP_DIR
                  partition p1 values less than (100) LOCATION
(''test1.csv''),
                   -- Use test2.csv in a specified directory MY_DIR
                  partition p2 values less than (300) DEFAULT DIRECTORY
MY_DIR LOCATION (''test2.csv'')        )'   );
END;
/
```

Example using the `file_uri_list` and `column_list` parameters with unstructured data files:

```
BEGIN
  DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
   table_name => 'MYSALES',
   credential_name => 'DEF_CRED_NAME',
   file_uri_list     => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/*.csv',
   column_list       => 'product varchar2(100), units number, country varchar2(100), year
number, month varchar2(2)',
   field_list        => 'product, units', --[Because country, year and month are not in the
file, they are not listed in the field list]
   format            => '{"type":"csv", "partition_columns":["country","year","month"]}');
END;
/
```

Example using the `file_uri_list` without the `column_list` parameter with structured data files:

```
BEGIN
  DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
  table_name => 'MYSALES',
  credential_name => 'DEF_CRED_NAME',
  DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
```

```
    table_name      => 'MYSALES',
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/*.parquet',
    format          =>
        json_object('type' value 'parquet', 'schema' value 'first',
                    'partition_columns' value
                          json_array(
                                json_object('name' value 'country', 'type' value
'varchar2(100)'),
                                json_object('name' value 'year', 'type' value 'number'),
                                json_object('name' value 'month', 'type' value 'varchar2(2)')
                          )
         )
    );
END;
/
```

CREATE_EXTERNAL_TABLE Procedure

This procedure creates an external table on files in the Cloud or from files in a directory. This allows you to run queries on external data from Autonomous Database.

Syntax

```
DBMS_CLOUD.CREATE_EXTERNAL_TABLE (
    table_name       IN VARCHAR2,
    credential_name  IN VARCHAR2 DEFAULT NULL,
    file_uri_list    IN CLOB,
    column_list      IN CLOB,
    field_list       IN CLOB DEFAULT,
    format           IN CLOB DEFAULT);
```

Parameters

Parameter	Description
`table_name`	The name of the external table.
`credential_name`	The name of the credential to access the Cloud Object Storage. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. This parameter is not used when you specify a directory with `file_uri_list`.

Parameter	Description
file_uri_list	This parameter specifies either a comma-delimited list of source file URIs or one or more directories and source files. **Cloud source file URIs** You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the regexuri format parameter is set to TRUE. The characters "*" and "?" are considered wildcard characters when the regexuri parameter is set to FALSE. When the regexuri parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the REGEXP_LIKE function. This option is only supported with external tables that are created on a file in the Object Storage. For example: `file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/)*[a-z]+[1-3]??.csv'` The format of the URIs depends on the Cloud Object Storage service you are using, for details see DBMS_CLOUD URI Formats. See REGEXP_LIKE Condition for more information on REGEXP_LIKE condition. **Directory** You can specify one directory and one or more file names or use a comma separated list of directories and file names. The format to specify a directory is:'*MY_DIR*:*filename.ext*'. By default the directory name *MY_DIR* is a database object and is case-insensitive. The file name is case sensitive. Regular expressions are not supported when specifying the file names in a directory. You can only use wildcards to specify file names in a directory. The character "*" can be used as the wildcard for multiple characters, and the character "?" can be used as the wildcard for a single character. For example:'*MY_DIR*:*" or '*MY_DIR*:test?' To specify multiple directories, use a comma separated list of directories: For example:'*MY_DIR1*:*, *MY_DIR2*:test?' Use double quotes to specify a case-sensitive directory name. For example:'"*my_dir1*":*, "*my_dir2*":Test?' To include a quote character, use two quotes. For example:'*MY_DIR*:''filename.ext'. This specifies the filename starts with a quote (').
column_list	Comma-delimited list of column names and data types for the external table.

Parameter	Description
field_list	Identifies the fields in the source files and their data types. The default value is NULL meaning the fields and their data types are determined by the column_list parameter. This argument's syntax is the same as the field_list clause in regular Oracle Database external tables. For more information about field_list see ORACLE_LOADER Access Driver field_list under field_definitions Clause in *Oracle Database Utilities*.
format	The options describing the format of the source files. For the list of the options and how to specify the values see DBMS_CLOUD Package Format Options. For Avro, ORC, or Parquet format files, see CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files.

Usage Notes

- The procedure DBMS_CLOUD.CREATE_EXTERNAL_TABLE supports external partitioned files in the supported cloud object storage services, including:
  - Oracle Cloud Infrastructure Object Storage
  - Azure Blob Storage
  - Amazon S3
  - Amazon S3-Compatible, including: Oracle Cloud Infrastructure Object Storage, Google Cloud Storage, and Wasabi Hot Cloud Storage.
  - GitHub Repository

  The credential is a table level property; therefore, the external files must be on the same object store.

  See DBMS_CLOUD URI Formats for more information.
- The default record delimiter is detected newline. With detected newline, DBMS_CLOUD tries to automatically find the correct newline character to use as the record delimiter. DBMS_CLOUD first searches for the Windows newline character \r\n. If it finds the Windows newline character, this is used as the record delimiter for all files in the procedure. If a Windows newline character is not found, DBMS_CLOUD searches for the UNIX/Linux newline character \n, and if it finds one it uses \n as the record delimiter for all files in the procedure. If the source files use a combination of different record delimiters, you may encounter an error such as, "KUP-04020: found record longer than buffer size supported". In this case, you need to either modify the source files to use the same record delimiter or only specify the source files that use the same record delimiter.

  See DBMS_CLOUD Package Format Options for information on the recorddelimiter format option.

Example

```
BEGIN
   DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
      table_name =>'WEATHER_REPORT_DOUBLE_DATE',
      credential_name =>'OBJ_STORE_CRED',
      file_uri_list =>'&base_URL/
Charlotte_NC_Weather_History_Double_Dates.csv',
```

```
      format => json_object('type' value 'csv', 'skipheaders' value '1'),
      field_list => 'REPORT_DATE DATE''mm/dd/yy'',
                     REPORT_DATE_COPY DATE ''yyyy-mm-dd'',
                     ACTUAL_MEAN_TEMP,
                     ACTUAL_MIN_TEMP,
                     ACTUAL_MAX_TEMP,
                     AVERAGE_MIN_TEMP,
                     AVERAGE_MAX_TEMP,
                     AVERAGE_PRECIPITATION',
      column_list => 'REPORT_DATE DATE,
                     REPORT_DATE_COPY DATE,
                     ACTUAL_MEAN_TEMP NUMBER,
                     ACTUAL_MIN_TEMP NUMBER,
                     ACTUAL_MAX_TEMP NUMBER,
                     AVERAGE_MIN_TEMP NUMBER,
                     AVERAGE_MAX_TEMP NUMBER,
                     AVERAGE_PRECIPITATION NUMBER');
   END;
/

SELECT * FROM WEATHER_REPORT_DOUBLE_DATE where
   actual_mean_temp > 69 and actual_mean_temp < 74
```

CREATE_EXTERNAL_TABLE Procedure for Apache Iceberg

This procedure creates external tables for Apache Iceberg tables in the supported configurations.

For a description of supported configurations, see About Querying Apache Iceberg Tables.

Syntax

```
DBMS_CLOUD.CREATE_EXTERNAL_TABLE (
    table_name       IN VARCHAR2,
    credential_name  IN VARCHAR2 DEFAULT NULL,
    file_uri_list    IN CLOB,
    column_list      IN CLOB DEFAULT NULL,
    field_list       IN CLOB DEFAULT NULL,
    format           IN CLOB DEFAULT NULL
);
```

Parameters

Parameter	Description
`table_name`	The name of the external table.
`credential_name`	The name of the credential used to access the data files, the metadata files and the Iceberg Catalog (if used). For AWS and OCI configurations, the credential should be created as described in CREATE_CREDENTIAL Procedure. AWS Amazon Resource Names (ARN) credentials are currently not supported.
`file_uri_list`	Must be NULL if an Iceberg catalog is specified (see `format` parameter below). If an iceberg catalog is not used, then the `file_uri_list` must contain the URI to the iceberg metadata file.

Parameter	Description
column_list	Must be NULL, as the column names and types are automatically derived from Iceberg metadata. The column names match the names found in the underlying data files (Parquet, Avro, ORC). The Oracle data types are derived using the Parquet/Avro/ORC mappings between Iceberg and the Parquet, Avro and ORC data types. Therefore users cannot specify the column_list.
field_list	Must be NULL, as column names and data types are automatically derived from the Iceberg metadata.
format	The format parameter has a different structure depending on the type of Iceberg table, AWS or OCI, and what information is used to create the external table, for example information from a data catalog or a direct metadata URI. For examples and further information: see the examples below, Iceberg Support on OCI Data Flow Samples, DBMS_CLOUD URI Formats.

Example AWS Iceberg tables using an AWS Glue Catalog

The format parameter when creating tables over an AWS Iceberg table using an AWS Glue Catalog is as follows:

```
format => json_object('access_protocol' value
       json_object('protocol_type' value 'iceberg',
                   'protocol_config' value
                    json_object('iceberg_catalog_type' value 'aws_glue',
                                'iceberg_glue_region'  value 'glue region',
                                'iceberg_table_path'   value
'database_name.table_name')));
```

Where, the access_protocol parameter contains a JSON object with two elements as follows:

- protocol_type: Must be 'iceberg'
- protocol_config: A nested JSON object specifying the iceberg catalog details.
  - iceberg_catalog_type: Must be 'aws_glue'
  - iceberg_glue_region: The catalog region, e.g. 'us-west-1'
  - iceberg_table_path: A *glue database.glue table name* path.

Example AWS Iceberg table using a metadata file URI

The format parameter when creating tables over an AWS Iceberg table using a metadata file URI, is as follows:

```
format => json_object('access_protocol' value
       json_object('protocol_type' value 'iceberg')
```

Example OCI Iceberg table using HadoopCatalog catalog

The `format` parameter when creating tables over an OCI Iceberg table created by OCI Data Flow using HadoopCatalog catalog, is as follows:

```
format => json_object('access_protocol' value
       json_object('protocol_type'   value 'iceberg',
                   'protocol_config' value
                   json_object('iceberg_catalog_type'  value 'hadoop',
                               'iceberg_warehouse'     value '<OCI folder
URI>',
                               'iceberg_table_path'    value
'database_name.table_name')));
```

Where, the `access_protocol` parameter contains a JSON object with two elements as follows:

- `protocol_type`: Must be `'iceberg'`
- `protocol_config`: A nested JSON object specifying the iceberg catalog details.
  - `iceberg_catalog_type`: Must be `'hadoop'`
  - `iceberg_warehouse`: The warehouse directory path used when generating the table, in native URI format.
  - `iceberg_table_path`: The *`database_name.table name`* path used when creating the table.

Example OCI Iceberg table using the URI of the metadata file

The `format` parameter when creating tables over an OCI Iceberg table using the URI of the metadata file, is as follows:

```
format => json_object('access_protocol' value
       json_object('protocol_type' value 'iceberg')
```

Where, the `access_protocol` parameter contains a JSON object with one element as follows:

- `protocol_type`: Must be `'iceberg'`

CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files

This procedure with the `format` parameter `type` set to the value `avro`, `orc`, or `parquet` creates an external table with either Avro, ORC, or Parquet format files in the Cloud or in a directory.

This allows you to run queries on external data from Autonomous Database.

Syntax

```
DBMS_CLOUD.CREATE_EXTERNAL_TABLE (
    table_name       IN VARCHAR2,
    credential_name  IN VARCHAR2 DEFAULT NULL,
    file_uri_list    IN CLOB,
    column_list      IN CLOB,
    field_list       IN CLOB DEFAULT,
    format           IN CLOB DEFAULT);
```

Parameters

Parameter	Description
`table_name`	The name of the external table.
`credential_name`	The name of the credential to access the Cloud Object Storage. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. This parameter is not used when you specify a directory with `file_uri_list`.

Parameter	Description
file_uri_list	This parameter specifies either a comma-delimited list of source file URIs or one or more directories and source files. **Cloud source file URIs** You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the regexuri format parameter is set to TRUE. The characters "*" and "?" are considered wildcard characters when the regexuri parameter is set to FALSE. When the regexuri parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the REGEXP_LIKE function. This option is only supported with external tables that are created on a file in the Object Storage. For example: `file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/)*[a-z]+[1-3]??.parquet'` The format of the URIs depends on the Cloud Object Storage service you are using, for details see DBMS_CLOUD URI Formats. See REGEXP_LIKE Condition for more information on REGEXP_LIKE condition. **Directory** You can specify one directory and one or more file names or use a comma separated list of directories and file names. The format to specify a directory is:'*MY_DIR*:*filename.ext*'. By default the directory name *MY_DIR* is a database object and is case-insensitive. The file name is case sensitive. Regular expressions are not supported when specifying the file names in a directory. You can only use wildcards to specify file names in a directory. The character "*" can be used as the wildcard for multiple characters, and the character "?" can be used as the wildcard for a single character. For example:'*MY_DIR*:*" or '*MY_DIR*:test?' To specify multiple directories, use a comma separated list of directories: For example:'*MY_DIR1*:*, *MY_DIR2*:test?' Use double quotes to specify a case-sensitive directory name. For example:'"*my_dir1*":*, "*my_dir2*":Test?' To include a quote character, use two quotes. For example:'*MY_DIR*:''filename.ext'. This specifies the filename starts with a quote ().

Parameter	Description
column_list	(Optional) This field, when specified, overrides the format->schema parameter which specifies that the schema, columns, and data types, are derived automatically. See the format parameter for details. When the column_list is specified for Avro, ORC, or Parquet source, the column names must match those columns found in the file. Oracle data types must map appropriately to the Avro, ORC, or Parquet data types. For Parquet files, see DBMS_CLOUD Package Parquet to Oracle Data Type Mapping for details. For ORC files, see DBMS_CLOUD Package ORC to Oracle Data Type Mapping for details. For Avro files, see DBMS_CLOUD Package Avro to Oracle Data Type Mapping for details.
field_list	Ignored for Avro, ORC, or Parquet files. The fields in the source match the external table columns by name. Source data types are converted to the external table column data type. For ORC files, see DBMS_CLOUD Package ORC to Oracle Data Type Mapping For Parquet files, see DBMS_CLOUD Package Parquet to Oracle Data Type Mapping for details. For Avro files, see DBMS_CLOUD Package Avro to Oracle Data Type Mapping for details.
format	For Avro, ORC, or Parquet type source files, see DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet for details.

Examples ORC

```
format => '{"type":"orc", "schema": "all"}'
```

```
format => json_object('type' value 'orc', 'schema' value 'first')
```

Examples Avro

```
format => '{"type":"avro", "schema": "all"}'
```

```
format => json_object('type' value 'avro', 'schema' value 'first')
```

Examples Parquet

```
format => '{"type":"parquet", "schema": "all"}'
```

```
format => json_object('type' value 'parquet', 'schema' value 'first')
```

Avro, ORC, or Parquet Column Name Mapping to Oracle Column Names

See DBMS_CLOUD Package Avro, ORC, and Parquet to Oracle Column Name Mapping for information on column name mapping and column name conversion usage in Oracle SQL.

CREATE_EXTERNAL_TEXT_INDEX Procedure

This procedure creates a text index on Object Storage files.

The `CREATE_EXTERNAL_TEXT_INDEX` procedure creates text index on the Object Storage files specified at the `location_uri` location. The index is refreshed at regular intervals, for any new additions or deletions done with files on location URI.

Syntax

```
DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX (
      credential_name  IN  VARCHAR2 DEFAULT NULL,
      location_uri     IN  VARCHAR2,
      index_name       IN  VARCHAR2,
      format           IN  CLOB     DEFAULT NULL
);
```

Parameters

Parameter	Description
`credential_name`	The name of the credential to access the Cloud Object Storage location. For public, pre-authenticated, or pre-signed bucket URIs, a NULL can be specified. See Configure Policies and Roles to Access Resources for more information. If you do not supply a `credential_name` value, the `credential_name` is set to a `NULL` value.
`location_uri`	Specifies the Object Store bucket or folder URI. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD URI Formats for more information.
`index_name`	Specifies the name of the index you are building on the files located at the `location_uri` location. This parameter is mandatory.

Parameter	Description
format	Specifies additional configuration options. Options are specified as a JSON string. The supported format options are: refresh_rate: Specifies the frequency in minutes at which the local index is refreshed. New file uploads and deletions result in an index refresh. The default value is 5 minutes. binary_files: Specifies if the contents of the files to be indexed are binary. For example, PDF, MS-Word, The default value is FALSE. stop_words: Specifies a list of stop words can be supplied when you create indexes. The stop_words value indicates if it is a list of stop words or a table of stop words. When a JSON array is provided the stop words parameter it is treated as a list, otherwise the stop words parameter is treated as a table name whose column "STOP_WORDS" is used to read in the list of stop words. You can specify stop words using the following methods: • JSON Array: For example: format := '{"stop_words": ["king","queen"]}' • Stop word table name: For example: format := '{"stop_words":"STOP_WORDS_TABLE"}' If you do not supply a format parameter, the format is set to a NULL value.

Example

```
BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX (
        credential_name => 'DEFAULT_CREDENTIAL',
        location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/ts_data/'
        index_name      => 'EMP',
        format          => JSON_OBJECT ('refresh_rate' value 10)
);
END;
/
```

CREATE_HYBRID_PART_TABLE Procedure

This procedure creates a hybrid partitioned table. This allows you to run queries on hybrid partitioned data from Autonomous Database using database objects and files in the Cloud, or database objects and files in a directory.

Syntax

```
DBMS_CLOUD.CREATE_HYBRID_PART_TABLE (
    table_name           IN VARCHAR2,
    credential_name      IN VARCHAR2,
    partitioning_clause  IN CLOB,
    column_list          IN CLOB,
    field_list           IN CLOB DEFAULT,
    format               IN CLOB DEFAULT);
```

Parameters

Parameter	Description
table_name	The name of the external table.
credential_name	The name of the credential to access the Cloud Object Storage. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information.
partitioning_clause	Specifies the complete partitioning clause, including the location information for individual partitions. To use directories, the partitioning clause supports the LOCATION and DEFAULT DIRECTORY values. You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the regexuri format parameter is set to TRUE. The characters "*" and "?" are considered wildcard characters when the regexuri parameter is set to FALSE. When the regexuri parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the REGEXP_LIKE function. Regular expression patterns are not supported for directory names. For example: `partitioning_clause => 'partition by range (col1) (partition p1 values less than (1000) external location ( ''https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/)*[a-z]+[1-3]??.txt''),….` See REGEXP_LIKE Condition for more information on REGEXP_LIKE condition.
column_list	Comma-delimited list of column names and data types for the external table.
field_list	Identifies the fields in the source files and their data types. The default value is NULL meaning the fields and their data types are determined by the *column_list* parameter. This argument's syntax is the same as the field_list clause in regular Oracle external tables. For more information about field_list see *Oracle® Database Utilities*.
format	The options describing the format of the source files. For the list of the options and how to specify the values see DBMS_CLOUD Package Format Options.

Usage Notes

- The procedure DBMS_CLOUD.CREATE_HYBRID_PART_TABLE supports external partitioned files in the supported cloud object storage services, including:
  - Oracle Cloud Infrastructure Object Storage

  - Azure Blob Storage
  - Amazon S3
  - Amazon S3-Compatible, including: Oracle Cloud Infrastructure Object Storage, Google Cloud Storage, and Wasabi Hot Cloud Storage.
  - GitHub Repository

  The credential is a table level property; therefore, the external files must be on the same object store.

  See DBMS_CLOUD URI Formats for more information.

- The procedure `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` supports hybrid partitioned files in directories, either in a local file system or in a network file system.
- The external partitioned tables you create with `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from.
  - `file$path`: Specifies the file path text up to the beginning of the object name.
  - `file$name`: Specifies the object name, including all the text that follows the bucket name.

Examples

```
BEGIN
   DBMS_CLOUD.CREATE_HYBRID_PART_TABLE(
      table_name =>'HPT1',
      credential_name =>'OBJ_STORE_CRED',
      format => json_object('delimiter' value ',', 'recorddelimiter' value
'newline', 'characterset' value 'us7ascii'),
      column_list => 'col1 number, col2 number, col3 number',
      partitioning_clause => 'partition by range (col1)
                                (partition p1 values less than (1000)
external location
                                    ( ''&base_URL/file_11.txt'')
                                ,
                                partition p2 values less than (2000)
external location
                                    ( ''&base_URL/file_21.txt'')
                                ,
                                partition p3 values less than (3000)
                                )'
     );
   END;
/


BEGIN
   DBMS_CLOUD.CREATE_HYBRID_PART_TABLE(
    table_name  => 'HPT1',
    format      => json_object('delimiter'value ',', 'recorddelimiter'value
'newline'),
    column_list => 'NAME VARCHAR2(30), GENDER VARCHAR2(10), BALANCE number',
    partitioning_clause => 'partition by range (B  2  ALANCE)
                (partition p1 values less than (1000) external DEFAULT
DIRECTORY DATA_PUMP_DIR LOCATION (''Scott_male_1000.csv''),
```

```
                partition p2 values less than (2000) external DEFAULT
DIRECTORY DATA_PUMP_DIR LOCATION (''Mary_female_3000.csv''),
                partition p3 values less than (3000))' );
END;
/
```

DELETE_ALL_OPERATIONS Procedure

This procedure clears either all data load operations logged in the `user_load_operations` table in your schema or clears all the data load operations of the specified type, as indicated with the `type` parameter.

Syntax

```
DBMS_CLOUD.DELETE_ALL_OPERATIONS (
    type      IN VARCHAR DEFAULT NULL);
```

Parameters

Parameter	Description
`type`	Specifies the type of operation to delete. Type values can be found in the `TYPE` column in the `user_load_operations` table. If no `type` is specified all rows are deleted.

Usage Note

- `DBMS_CLOUD.DELETE_ALL_OPERATIONS` does not delete currently running operations (operations in a "Running" status).

DELETE_FILE Procedure

This procedure removes the specified file from the specified directory on Autonomous Database.

Syntax

```
DBMS_CLOUD.DELETE_FILE (
      directory_name     IN VARCHAR2,
      file_name          IN VARCHAR2,
      force              IN BOOLEAN DEFAULT FALSE);
```

Parameters

Parameter	Description
`directory_name`	The name of the directory on the Autonomous Database instance.
`file_name`	The name of the file to be removed.
`force`	Ignore and do not report errors if the file does not exist. Valid values are: `TRUE` and `FALSE`. The default value is `FALSE`.

> **Note:**
>
> To run `DBMS_CLOUD.DELETE_FILE` with a user other than ADMIN you need to grant write privileges on the directory that contains the file to that user. For example, run the following command as ADMIN to grant write privileges to `adb_user`:
>
> ```
> GRANT WRITE ON DIRECTORY data_pump_dir TO adb_user;
> ```

Example

```
BEGIN
   DBMS_CLOUD.DELETE_FILE(
      directory_name =>  'DATA_PUMP_DIR',
      file_name => 'exp1.dmp' );
   END;
/
```

DELETE_OBJECT Procedure

This procedure deletes the specified object on object store.

Syntax

```
DBMS_CLOUD.DELETE_OBJECT (
       credential_name      IN VARCHAR2,
       object_uri           IN VARCHAR2,
       force                IN BOOLEAN DEFAULT FALSE);
```

Parameters

Parameter	Description
`credential_name`	The name of the credential to access the Cloud Object Storage. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information.
`object_uri`	Object or file URI for the object to delete. The format of the URI depends on the Cloud Object Storage service you are using, for details see DBMS_CLOUD URI Formats.
`force`	Ignore and do not report errors if object does not exist. Valid values are: `TRUE` and `FALSE`. The default value is `FALSE`.

Example

```
BEGIN
   DBMS_CLOUD.DELETE_OBJECT(
       credential_name => 'DEF_CRED_NAME',
       object_uri => 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/exp1.dmp' );
```

```
   END;
/
```

DROP_EXTERNAL_TEXT_INDEX Procedure

This procedure drops text index on the Object Storage files.

The `DROP_EXTERNAL_TEXT_INDEX` procedure drops the specified index created with the `CREATE_EXTERNAL_TEXT_INDEX` procedure.

Syntax

```
DBMS_CLOUD.DROP_EXTERNAL_TEXT_INDEX (
      index_name       IN  VARCHAR2,
);
```

Parameters

Parameter	Description
`index_name`	Specifies the name of the index you are dropping. The index name must match the name provided at the time of the index creation. This parameter is mandatory.

Example

```
BEGIN
DBMS_CLOUD.DROP_EXTERNAL_TEXT_INDEX (
        index_name => 'EMP',
);
END;
/
```

EXPORT_DATA Procedure

This procedure exports data from Autonomous Database based on the result of a query. This procedure is overloaded and supports writing files to the cloud or to a directory.

Based on the `format type` parameter, the procedure exports files to the Cloud or to a directory location as text files in CSV, JSON, Parquet, or XML format, or using the ORACLE_DATAPUMP access driver to write data to an Oracle Datapump dump file.

Syntax

```
DBMS_CLOUD.EXPORT_DATA (
      file_uri_list     IN CLOB,
      format            IN CLOB,
      credential_name   IN VARCHAR2 DEFAULT NULL,
      query             IN CLOB);

DBMS_CLOUD.EXPORT_DATA (
      file_uri_list     IN CLOB DEFAULT NULL,
      format            IN CLOB DEFAULT NULL,
```

```
    credential_name   IN VARCHAR2 DEFAULT NULL,
    query             IN CLOB DEFAULT NULL,
    operation_id      OUT NOCOPY NUMBER);
```

Parameters

Parameter	Description
credential_name	The name of the credential to access the Cloud Object Storage. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. When the credential parameter is not included, this specifies output to a directory.
file_uri_list	There are different forms, depending on the value of the format parameter and depending on whether you include a credential parameter: • When the format parameter type value is json: The JSON on Object Store or to the specified directory location is saved with a generated file name based on the value of the file_uri_list parameter. See File Naming for Text Output (CSV, JSON, Parquet, or XML) for more information. • When the format parameter type value is datapump, the file_uri_list is a comma-delimited list of the dump files. This specifies the files to be created on the Object Store. Use of wildcard and substitution characters is not supported in the file_uri_list. • When the credential_name parameter is not specified you provide a directory name in file_uri_list. The format of the URIs depend on the Cloud Object Storage service you are using, for details see DBMS_CLOUD URI Formats.
format	A JSON string that provides export format options. Supported option is: • type: The type format option is required and must have one of the values: csv | datapump | json | parquet | xml. See DBMS_CLOUD Package Format Options for EXPORT_DATA.
query	Use this parameter to specify a SELECT statement so that only the required data is exported. The query determines the contents of the files you export as text files CSV, JSON, Parquet, or XML, or as dump files. For example: SELECT warehouse_id, quantity FROM inventories For information with the format type value datapump, see Oracle Data Pump Export Data Filters and Unloading and Loading Data with the ORACLE_DATAPUMP Access Driver for more information. When the format type value is json, each query result is checked and if it is not JSON, as determined with the function: JSON_OBJECT_T.parse(), DBMS_CLOUD.EXPORT_DATA transforms the query to include JSON_OBJECT function to convert the row into JSON. See JSON_OBJECT_T Object Type for more information. For example: SELECT JSON_OBJECT(* RETURNING CLOB) from(SELECT warehouse_id, quantity FROM inventories)
operation_id	Use this parameter to track the progress and final status of the export operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Usage Notes:

- The `query` parameter value that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Depending on the format parameter specified, `DBMS_CLOUD.EXPORT_DATA` outputs the results of the specified query on the Cloud Object Store or to a directory location in one of these formats:
  - CSV, JSON, Parquet, or XML files.

    See Export Data to Object Store as Text and Export Data to a Directory for more information on using `DBMS_CLOUD.EXPORT_DATA` with CSV, JSON, Parquet, or XML output files.
  - Using the ORACLE_DATAPUMP access driver to write data to a dump file.
- For CSV, JSON, or XML output, by default when a generated file contains 10MB of data a new output file is created. However, if you have less than 10MB of result data you may have multiple output files, depending on the database service and the number of ECPUs (OCPUs if your database uses OCPUs) for the Autonomous Database instance.

  See File Naming for Text Output (CSV, JSON, Parquet, or XML) for more information.

  The default output file chunk size is 10MB for CSV, JSON, or XML. You can change this value with the `format` parameter `maxfilesize` option. See DBMS_CLOUD Package Format Options for EXPORT_DATA for more information.
- For Parquet output, each generated file is less than 128MB and multiple output files may be generated. However, if you have less than 128MB of result data, you may have multiple output files depending on the database service and the number of ECPUs (OCPUs if your database uses OCPUs) for the Autonomous Database instance.

  See File Naming for Text Output (CSV, JSON, Parquet, or XML) for more information.

Usage Notes for ORACLE_DATAPUMP Output (`DBMS_CLOUD.EXPORT_DATA` with `format` parameter `type` option `datapump`):

- `EXPORT_DATA` uses `DATA_PUMP_DIR` as the default logging directory. So the write privilege on `DATA_PUMP_DIR` is required when using `ORACLE_DATAPUMP` output.
- Autonomous Database export using `DBMS_CLOUD.EXPORT_DATA` with `format` parameter `type` option `datapump` only supports Oracle Cloud Infrastructure Object Storage, Oracle Cloud Infrastructure Object Storage Classic object stores or directory output.
- When you specify `DBMS_CLOUD.EXPORT_DATA` with the `format` parameter `type` option `datapump`, the `credential_name` parameter value cannot be an OCI resource principal.
- Oracle Data Pump divides each dump file part into smaller chunks for faster uploads. The Oracle Cloud Infrastructure Object Storage console shows multiple files for each dump file part that you export. The size of the actual dump files will be displayed as zero (0) and its related file chunks as 10mb or less. For example:

  ```
  exp01.dmp
  exp01.dmp_aaaaaa
  exp02.dmp
  exp02.dmp_aaaaaa
  ```

  Downloading the zero byte dump file from the Oracle Cloud Infrastructure console or using the Oracle Cloud Infrastructure CLI will not give you the full dump files. To download the

full dump files from the Object Store, use a tool that supports Swift such as curl, and provide your user login and Swift auth token.

```
curl -O -v -X GET -u 'user1@example.com:auth_token' \
   https://swiftobjectstorage.us-ashburn-1.oraclecloud.com/v1/namespace-
string/bucketname/exp01.dmp
```

If you import a file with the DBMS_CLOUD procedures that support the format parameter type with the value 'datapump', you only need to provide the primary file name. The procedures that support the 'datapump' format type automatically discover and download the chunks.

When you use DBMS_CLOUD.DELETE_OBJECT, the procedure automatically discovers and deletes the chunks when the procedure deletes the primary file.

- The DBMS_CLOUD.EXPORT_DATA procedure creates the dump file(s) from the file_uri_list values that you specify, as follows:
  - As more files are needed, the procedure creates additional files from the file_uri_list.
  - The procedure does not overwrite files. If a dump file in the file_uri_list exists, DBMS_CLOUD.EXPORT_DATA reports an error.
  - DBMS_CLOUD.EXPORT_DATA does not create buckets.
- The number of dump files that DBMS_CLOUD.EXPORT_DATA generates is determined when the procedure runs. The number of dump files that are generated depends on the number of file names you provide in the file_uri_list parameter, as well as on the number of Autonomous Database OCPUs available to the instance, the service level, and the size of the data.

  For example, if you use a 1 OCPU Autonomous Database instance or the low service, then a single dump file is exported with no parallelism, even if you provide multiple file names. If you use a 4 OCPU Autonomous Database instance with the medium or high service, then the jobs can run in parallel and multiple dump files are exported if you provide multiple file names.
- The dump files you create with DBMS_CLOUD.EXPORT_DATA cannot be imported using Oracle Data Pump impdp. Depending on the database, you can use these files as follows:
  - On an Autonomous Database, you can use the dump files with the DBMS_CLOUD procedures that support the format parameter type with the value 'datapump'. You can import the dump files using DBMS_CLOUD.COPY_DATA or you can call DBMS_CLOUD.CREATE_EXTERNAL_TABLE to create an external table.
  - On any other Oracle Database, such as Oracle Database 19c on-premise, you can import the dump files created with the procedure DBMS_CLOUD.EXPORT_DATA using the ORACLE_DATAPUMP access driver. See Unloading and Loading Data with the ORACLE_DATAPUMP Access Driver for more information.
- The query parameter value that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.

Usage Notes for DBMS_CLOUD.EXPORT_DATA with Output to a Directory

- The provided directory must exist and you must be logged in as the ADMIN user or have WRITE access to the directory.
- DBMS_CLOUD.EXPORT_DATA does not create directories.

- The procedure does not overwrite files. For example, if a dump file in the `file_uri_list` exists, `DBMS_CLOUD.EXPORT_DATA` reports an error such as:

```
ORA-31641: unable to create dump file  "/u02/exports/123.dmp"
ORA-27038: created file already exists
```

Examples

The following example shows `DBMS_CLOUD.EXPORT_DATA` with the format `type` parameter with the value `datapump`:

```
BEGIN
   DBMS_CLOUD.EXPORT_DATA(
      credential_name =>'OBJ_STORE_CRED',
      file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/exp1.dmp',
      format => json_object('type' value 'datapump', 'compression' value
'basic', 'version' value 'latest'),
      query => 'SELECT warehouse_id, quantity FROM inventories'
     );
   END;
/
```

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

The following example shows `DBMS_CLOUD.EXPORT_DATA` with the format `type` parameter with the value `json`:

```
BEGIN
   DBMS_CLOUD.EXPORT_DATA(
      credential_name => 'OBJ_STORE_CRED',
      file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp1.json',
      query           => 'SELECT * FROM DEPT',
      format          => JSON_OBJECT('type' value 'json', 'compression' value
'gzip'));
     );
   END;
/
```

The following example shows `DBMS_CLOUD.EXPORT_DATA` with the format `type` parameter with the value `xml`:

```
BEGIN
   DBMS_CLOUD.EXPORT_DATA(
      credential_name => 'OBJ_STORE_CRED',
      file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp1.xml',
      query           => 'SELECT * FROM DEPT',
      format          => JSON_OBJECT('type' value 'xml', 'compression' value
'gzip'));
```

```
     );
   END;
/
```

The following example shows `DBMS_CLOUD.EXPORT_DATA` with the format `type` parameter with the value `csv`:

```
BEGIN
   DBMS_CLOUD.EXPORT_DATA(
      credential_name => 'OBJ_STORE_CRED',
      file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp.csv',
      query           => 'SELECT * FROM DEPT',
      format          => JSON_OBJECT('type' value 'csv', 'delimiter' value
'|', 'compression' value 'gzip', 'header' value true, 'encryption' value
('user_defined_function' value 'ADMIN.decryption_callback')));
     );
   END;
/
```

The following example shows `DBMS_CLOUD.EXPORT_DATA` exporting data to a directory location with the `type` parameter with the value `datapump`:

```
BEGIN
 DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => 'export_dir:sales.dmp',
    format        => json_object('type' value 'datapump'),
    query         => 'SELECT * FROM sales'
 );
END;
/
```

GET_OBJECT Procedure and Function

This procedure is overloaded. The procedure form reads an object from Cloud Object Storage and copies it to Autonomous Database. The function form reads an object from Cloud Object Storage and returns a `BLOB` to Autonomous Database.

Syntax

```
DBMS_CLOUD.GET_OBJECT (
       credential_name      IN VARCHAR2,
       object_uri           IN VARCHAR2,
       directory_name       IN VARCHAR2,
       file_name            IN VARCHAR2 DEFAULT  NULL,
       startoffset          IN NUMBER DEFAULT  0,
       endoffset            IN NUMBER DEFAULT  0,
       compression          IN VARCHAR2 DEFAULT  NULL);


DBMS_CLOUD.GET_OBJECT(
       credential_name      IN VARCHAR2 DEFAULT NULL,
       object_uri           IN VARCHAR2,
       startoffset          IN NUMBER DEFAULT  0,
```

```
      endoffset             IN NUMBER DEFAULT  0,
      compression           IN VARCHAR2 DEFAULT  NULL)
RETURN BLOB;
```

Parameters

Parameter	Description
credential_name	The name of the credential to access the Cloud Object Storage. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information.
object_uri	Object or file URI. The format of the URI depends on the Cloud Object Storage service you are using, for details see DBMS_CLOUD URI Formats.
directory_name	The name of the directory on the database. 1
file_name	Specifies the name of the file to create. If file name is not specified, the file name is taken from after the last slash in the object_uri parameter. For special cases, for example when the file name contains slashes, use the file_name parameter.
startoffset	The offset, in bytes, from where the procedure starts reading.
endoffset	The offset, in bytes, until where the procedure stops reading.
compression	Specifies the compression used to store the object. When compression is set to 'AUTO' the file is uncompressed (the value 'AUTO' implies the object specified with object_uri is compressed with Gzip).

> **Note:**
>
> To run DBMS_CLOUD.GET_OBJECT with a user other than ADMIN you need to grant WRITE privileges on the directory to that user. For example, run the following command as ADMIN to grant write privileges to adb_user:
>
> ```
> GRANT WRITE ON DIRECTORY data_pump_dir TO adb_user;
> ```

Return Values

The function form reads from Object Store and DBMS_CLOUD.GET_OBJECT returns a BLOB.

Examples

```
BEGIN
   DBMS_CLOUD.GET_OBJECT(
     credential_name => 'OBJ_STORE_CRED',
     object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/file.txt',
     directory_name => 'DATA_PUMP_DIR');
END;
/
```

To read character data from a file in Object Store:

```
SELECT to_clob(
     DBMS_CLOUD.GET_OBJECT(
       credential_name => 'OBJ_STORE_CRED',
       object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/file.txt'))
FROM DUAL;
```

To add an image stored on Object Store in a `BLOB` in the database:

```
DECLARE
   l_blob BLOB := NULL;
BEGIN
   l_blob := DBMS_CLOUD.GET_OBJECT(
     credential_name => 'OBJ_STORE_CRED',
     object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/MyImage.gif' );
END;
/
```

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

LIST_FILES Function

This function lists the files in the specified directory. The results include the file names and additional metadata about the files such as file size in bytes, creation timestamp, and the last modification timestamp.

Syntax

```
DBMS_CLOUD.LIST_FILES (
    directory_name      IN VARCHAR2)
       RETURN TABLE;
```

Parameters

Parameter	Description
`directory_name`	The name of the directory on the database.

Usage Notes

- To run `DBMS_CLOUD.LIST_FILES` with a user other than ADMIN you need to grant read privileges on the directory to that user. For example, run the following command as ADMIN to grant read privileges to `adb_user`:

  ```
  GRANT READ ON DIRECTORY data_pump_dir TO adb_user;
  ```

- This is a pipelined table function with return type as `DBMS_CLOUD_TYPES.list_object_ret_t`.

- `DBMS_CLOUD.LIST_FILES` does not obtain the checksum value and returns `NULL` for this field.

Example

This is a pipelined function that returns a row for each file. For example, use the following query to use this function:

```
SELECT * FROM DBMS_CLOUD.LIST_FILES('DATA_PUMP_DIR');

OBJECT_NAME       BYTES   CHECKSUM      CREATED              LAST_MODIFIED
------------ ---------- ----------    ---------------------
---------------------
cwallet.sso        2965               2018-12-12T18:10:47Z
2019-11-23T06:36:54Z
```

LIST_OBJECTS Function

This function lists objects in the specified location on object store. The results include the object names and additional metadata about the objects such as size, checksum, creation timestamp, and the last modification timestamp.

Syntax

```
DBMS_CLOUD.LIST_OBJECTS (
       credential_name      IN VARCHAR2,
       location_uri         IN VARCHAR2)
   RETURN TABLE;
```

Parameters

Parameter	Description
`credential_name`	The name of the credential to access the Cloud Object Storage. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information.
`location_uri`	Object or file URI. The format of the URI depends on the Cloud Object Storage service you are using, for details see DBMS_CLOUD URI Formats.

Usage Notes

- Depending on the capabilities of the object store, `DBMS_CLOUD.LIST_OBJECTS` does not return values for certain attributes and the return value for the field is `NULL` in this case.

  All supported Object Stores return values for the `OBJECT_NAME`, `BYTES`, and `CHECKSUM` fields.

  The following table shows support for the fields `CREATED` and `LAST_MODIFIED` by Object Store:

Object Store	CREATED	LAST_MODIFIED
Oracle Cloud Infrastructure Native	Returns timestamp	Returns timestamp
Oracle Cloud Infrastructure Swift	Returns `NULL`	Returns timestamp

Object Store	CREATED	LAST_MODIFIED
Oracle Cloud Infrastructure Classic	Returns NULL	Returns timestamp
Amazon S3	Returns NULL	Returns timestamp
Amazon S3-Compatible	Returns NULL	Returns timestamp
Azure	Returns timestamp	Returns timestamp
GitHub Repository		

- The checksum value is the MD5 checksum. This is a 32-character hexadecimal number that is computed on the object contents. It is expected to have a different checksum value if OCI$RESOURCE_PRINCIPAL credential is used.
- This is a pipelined table function with return type as DBMS_CLOUD_TYPES.list_object_ret_t.

Example

This is a pipelined function that returns a row for each object. For example, use the following query to use this function:

```
SELECT * FROM DBMS_CLOUD.LIST_OBJECTS('OBJ_STORE_CRED',
    'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o/');


OBJECT_NAME   BYTES              CHECKSUM
CREATED         LAST_MODIFIED
------------ ---------- --------------------------------
-------------------- --------------------
cwallet.sso   2965      2339a2731ba24a837b26d344d643dc07
2019-11-23T06:36:54Z
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.

MOVE_OBJECT Procedure

This procedure moves an object from one Cloud Object Storage bucket or folder to another.

The source and target bucket or folder can be in the same or different Cloud Object store provider.

When the source and target are in distinct Object Stores or have different accounts with the same cloud provider, you can give separate credential names for the source and target locations.

The source credential name is by default also used by the target location when target credential name is not provided.

Syntax

```
DBMS_CLOUD.MOVE_OBJECT (
    source_credential_name  IN  VARCHAR2 DEFAULT NULL,
    source_object_uri       IN  VARCHAR2,
    target_object_uri       IN  VARCHAR2,
```

```
    target_credential_name  IN  VARCHAR2 DEFAULT NULL
);
```

Parameters

Parameter	Description
source_credential_name	The name of the credential to access the source Cloud Object Storage. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. If you do not supply a source_credential_name value, the credential_name is set to NULL.
source_object_uri	Specifies URI, that point to the source Object Storage bucket or folder location. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD URI Formats for more information.
target_object_uri	Specifies the URI for the target Object Storage bucket or folder, where the files need to be moved. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD URI Formats for more information.
target_credential_name	The name of the credential to access the target Cloud Object Storage location. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. If you do not supply a target_credential_name value, the target_object_uri is set to the source_credential_name value.

Example

```
BEGIN
DBMS_CLOUD.MOVE_OBJECT (
    source_credential_name => 'OCI_CRED',
    source_object_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/bgfile.csv',
    target_object_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/myfile.csv'
);
END;
/
```

PUT_OBJECT Procedure

This procedure is overloaded. In one form the procedure copies a file from Autonomous Database to the Cloud Object Storage. In another form the procedure copies a BLOB from Autonomous Database to the Cloud Object Storage.

Syntax

```
DBMS_CLOUD.PUT_OBJECT (
       credential_name      IN VARCHAR2,
       object_uri           IN VARCHAR2,
       directory_name       IN VARCHAR2,
       file_name            IN VARCHAR2);


DBMS_CLOUD.PUT_OBJECT (
       credential_name      IN VARCHAR2,
       object_uri           IN VARCHAR2,
       contents             IN BLOB,
       file_name            IN VARCHAR2);
```

Parameters

Parameter	Description
`credential_name`	The name of the credential to access the Cloud Object Storage. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information.
`object_uri`	Object or file URI. The format of the URI depends on the Cloud Object Storage service you are using, for details see DBMS_CLOUD URI Formats.
`directory_name`	The name of the directory on the Autonomous Database. [1]
`file_name`	The name of the file in the specified directory.

> **Note:**
>
> To run `DBMS_CLOUD.PUT_OBJECT` with a user other than ADMIN you need to grant read privileges on the directory to that user. For example, run the following command as ADMIN to grant read privileges to `adb_user`:
>
> ```
> GRANT READ ON DIRECTORY data_pump_dir TO adb_user;
> ```

Example

To handle `BLOB` data after in-database processing and then store the data directly into a file in the object store:

```
DECLARE
      my_blob_data BLOB;
BEGIN
 /* Some processing producing BLOB data and populating my_blob_data */
DBMS_CLOUD.PUT_OBJECT(
     credential_name => 'OBJ_STORE_CRED',
     object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/my_new_file',
```

```
    contents => my_blob_data));
END;
/
```

Usage Notes

Depending on your Cloud Object Storage, the size of the object you transfer is limited as follows:

Cloud Object Storage Service	Object Transfer Size Limit
Oracle Cloud Infrastructure Object Storage	50 GB
Amazon S3	5 GB
Azure Blob Storage	256 MB
Amazon S3-Compatible	Set by the object store provider. For more information, refer to the provider's documentation.

Oracle Cloud Infrastructure object store does not allow writing files into a public bucket without supplying credentials (Oracle Cloud Infrastructure allows users to download objects from public buckets). Thus, you must supply a credential name with valid credentials to store an object in an Oracle Cloud Infrastructure public bucket using `PUT_OBJECT`.

See DBMS_CLOUD URI Formats for more information.

SYNC_EXTERNAL_PART_TABLE Procedure

This procedure simplifies updating an external partitioned table from files in the Cloud. Run this procedure whenever new partitions are added or when partitions are removed from the Object Store source for the external partitioned table.

Syntax

```
DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE (
    table_name         IN VARCHAR2,
    schema_name        IN VARCHAR2 DEFAULT,
    update_columns     IN BOOLEAN DEFAULT);
```

Parameters

Parameter	Description
`table_name`	The name of the target table. The target table needs to be created before you run `DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE`.
`schema_name`	The name of the schema where the target table resides. The default value is NULL meaning the target table is in the same schema as the user running the procedure.
`update_columns`	The new files may introduce a change to the schema. Updates supported include: new columns, deleted columns. Updates to existing columns, for example a change in the data type throw errors. Default Value: False

VALIDATE_EXTERNAL_PART_TABLE Procedure

This procedure validates the source files for an external partitioned table, generates log information, and stores the rows that do not match the format options specified for the external table in a *badfile* table on Autonomous Database. The overloaded form enables you to use the `operation_id` parameter.

Syntax

```
DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE (
       table_name                 IN VARCHAR2,
       partition_name             IN CLOB DEFAULT,
       subpartition_name          IN CLOB DEFAULT,
       schema_name                IN VARCHAR2 DEFAULT,
       rowcount                   IN NUMBER DEFAULT,
       partition_key_validation   IN BOOLEAN DEFAULT,
       stop_on_error              IN BOOLEAN DEFAULT);


DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE (
       table_name                 IN VARCHAR2,
       operation_id               OUT NUMBER,
       partition_name             IN CLOB DEFAULT,
       subpartition_name          IN CLOB DEFAULT,
       schema_name                IN VARCHAR2 DEFAULT,
       rowcount                   IN NUMBER DEFAULT,
       partition_key_validation   IN BOOLEAN DEFAULT,
       stop_on_error              IN BOOLEAN DEFAULT);
```

Parameters

Parameter	Description
`table_name`	The name of the external table.
`operation_id`	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the `USER_LOAD_OPERATIONS` view.
`partition_name`	If defined, then only a specific partition is validated. If not specified then read all partitions sequentially until `rowcount` is reached.
`subpartition_name`	If defined, then only a specific subpartition is validated. If not specified then read from all external partitions or subpartitions sequentially until `rowcount` is reached.
`schema_name`	The name of the schema where the external table resides. The default value is NULL meaning the external table is in the same schema as the user running the procedure.
`rowcount`	Number of rows to be scanned. The default value is NULL meaning all the rows in the source files are scanned.
`partition_key_validation`	For internal use only. Do not use this parameter.
`stop_on_error`	Determines if the validate should stop when a row is rejected. The default value is `TRUE` meaning the validate stops at the first rejected row. Setting the value to `FALSE` specifies that the validate does not stop at the first rejected row and validates all rows up to the value specified for the `rowcount` parameter.

VALIDATE_EXTERNAL_TABLE Procedure

This procedure validates the source files for an external table, generates log information, and stores the rows that do not match the format options specified for the external table in a *badfile* table on Autonomous Database. The overloaded form enables you to use the `operation_id` parameter.

Syntax

```
DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE (
    table_name      IN VARCHAR2,
    schema_name     IN VARCHAR2 DEFAULT,
    rowcount        IN NUMBER DEFAULT,
    stop_on_error   IN BOOLEAN DEFAULT);


DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE(
    table_name      IN VARCHAR2,
    operation_id    OUT NOCOPY NUMBER,
    schema_name     IN VARCHAR2 DEFAULT NULL,
    rowcount        IN NUMBER DEFAULT 0,
    stop_on_error   IN BOOLEAN DEFAULT TRUE);
```

Parameters

Parameter	Description
`table_name`	The name of the external table.
`operation_id`	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the `USER_LOAD_OPERATIONS` view.
`schema_name`	The name of the schema where the external table resides. The default value is NULL meaning the external table is in the same schema as the user running the procedure.
`rowcount`	Number of rows to be scanned. The default value is NULL meaning all the rows in the source files are scanned.
`stop_on_error`	Determines if the validate should stop when a row is rejected. The default value is `TRUE` meaning the validate stops at the first rejected row. Setting the value to `FALSE` specifies that the validate does not stop at the first rejected row and validates all rows up to the value specified for the `rowcount` parameter. If the external table refers to Avro, ORC, or Parquet files then the validate stops at the first rejected row. When the external table specifies the `format` parameter `type` set to the value `avro`, `orc`, or `parquet`, the parameter `stop_on_error` effectively always has the value `TRUE`. Thus, the table badfile will always be empty for an external table referring to Avro, ORC, or Parquet files.

Usage Notes

- `DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE` works with both partitioned external tables and hybrid partitioned tables. This potentially reads data from all external partitions until `rowcount` is reached or `stop_on_error` applies. You do not have control over which partition, or parts of a partition, is read in which order.

VALIDATE_HYBRID_PART_TABLE Procedure

This procedure validates the source files for a hybrid partitioned table, generates log information, and stores the rows that do not match the format options specified for the hybrid table in a *badfile* table on Autonomous Database. The overloaded form enables you to use the `operation_id` parameter.

Syntax

```
DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE (
       table_name                 IN VARCHAR2,
       partition_name             IN CLOB DEFAULT,
       subpartition_name          IN CLOB DEFAULT,
       schema_name                IN VARCHAR2 DEFAULT,
       rowcount                   IN NUMBER DEFAULT,
       partition_key_validation   IN BOOLEAN DEFAULT,
       stop_on_error              IN BOOLEAN DEFAULT);


DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE (
       table_name                 IN VARCHAR2,
       operation_id               OUT NUMBER,
       partition_name             IN CLOB DEFAULT,
       subpartition_name          IN CLOB DEFAULT,
       schema_name                IN VARCHAR2 DEFAULT,
       rowcount                   IN NUMBER DEFAULT,
       partition_key_validation   IN BOOLEAN DEFAULT,
       stop_on_error              IN BOOLEAN DEFAULT);
```

Parameters

Parameter	Description
`table_name`	The name of the external table.
`operation_id`	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the `USER_LOAD_OPERATIONS` view.
`partition_name`	If defined, then only a specific partition is validated. If not specified then read from all external partitions sequentially until `rowcount` is reached.
`subpartition_name`	If defined, then only a specific subpartition is validated. If not specified then read from all external partitions or subpartitions sequentially until `rowcount` is reached.
`schema_name`	The name of the schema where the external table resides. The default value is NULL meaning the external table is in the same schema as the user running the procedure.
`rowcount`	Number of rows to be scanned. The default value is NULL meaning all the rows in the source files are scanned.
`partition_key_validation`	For internal use only. Do not use this parameter.
`stop_on_error`	Determines if the validate should stop when a row is rejected. The default value is `TRUE` meaning the validate stops at the first rejected row. Setting the value to `FALSE` specifies that the validate does not stop at the first rejected row and validates all rows up to the value specified for the `rowcount` parameter.

DBMS_CLOUD for Bulk File Management

The subprograms for bulk file operations within the DBMS_CLOUD package.

Subprogram	Description
BULK_COPY Procedure	This procedure copies files from one Cloud Object Storage bucket to another.
BULK_DELETE Procedure	The procedure deletes files from Cloud Object Storage bucket or folder.
BULK_DOWNLOAD Procedure	This procedure downloads files from Cloud Object store bucket to a directory in Autonomous Database.
BULK_MOVE Procedure	This procedure moves files from one Cloud Object Storage bucket to another.
BULK_UPLOAD Procedure	This procedure uploads files from a directory in Autonomous Database to the Cloud Object Storage.

- BULK_COPY Procedure
  This procedure bulk copies files from one Cloud Object Storage bucket to another. The overloaded form enables you to use the `operation_id` parameter.
- BULK_DELETE Procedure
  This procedure bulk deletes files from the Cloud Object Storage. The overloaded form enables you to use the `operation_id` parameter. You can filter the list of files to be deleted using a regular expression pattern compatible with `REGEXP_LIKE` operator.
- BULK_DOWNLOAD Procedure
  This procedure downloads files into an Autonomous Database directory from Cloud Object Storage. The overloaded form enables you to use the `operation_id` parameter. You can filter the list of files to be downloaded using a regular expression pattern compatible with `REGEXP_LIKE` operator.
- BULK_MOVE Procedure
  This procedure bulk moves files from one Cloud Object Storage bucket or folder to another. The overloaded form enables you to use the `operation_id` parameter.
- BULK_UPLOAD Procedure
  This procedure copies files into Cloud Object Storage from an Autonomous Database directory. The overloaded form enables you to use the `operation_id` parameter.

BULK_COPY Procedure

This procedure bulk copies files from one Cloud Object Storage bucket to another. The overloaded form enables you to use the `operation_id` parameter.

You can filter the list of files to be deleted using a regular expression pattern compatible with `REGEXP_LIKE` operator.

The source and target bucket or folder can be in the same or different Cloud Object store provider.

When the source and target are in distinct Object Stores or have different accounts with the same cloud provider, you can give separate credential names for the source and target locations.

The source credential name is by default also used by the target location.

Syntax

```
DBMS_CLOUD.BULK_COPY (
      source_credential_name  IN  VARCHAR2 DEFAULT NULL,
      source_location_uri     IN  VARCHAR2,
      target_location_uri     IN  VARCHAR2,
      target_credential_name  IN  VARCHAR2 DEFAULT NULL,
      regex_filter            IN  VARCHAR2 DEFAULT NULL,
      format                  IN  CLOB     DEFAULT NULL
);

DBMS_CLOUD.BULK_COPY (
      source_credential_name  IN  VARCHAR2 DEFAULT NULL,
      source_location_uri     IN  VARCHAR2,
      target_location_uri     IN  VARCHAR2,
      target_credential_name  IN  VARCHAR2 DEFAULT NULL,
      regex_filter            IN  VARCHAR2 DEFAULT NULL,
      format                  IN  CLOB     DEFAULT NULL,
      operation_id            OUT NUMBER
);
```

Parameters

Parameter	Description
`source_credential_name`	The name of the credential to access the Cloud Object Storage. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. If you do not supply a `source_credential_name` value, the `credential_name` is set to `NULL`.
`source_location_uri`	Specifies URI, that point to the source Object Storage bucket or folder location. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD URI Formats for more information.
`target_location_uri`	Specifies the URI for the target Object Storage bucket or folder, where the files need to be copied. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD URI Formats for more information.
`target_credential_name`	The name of the credential to access the target Cloud Object Storage location. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. If you do not supply a `target_credential_name` value, the `target_location_uri` is set to the `source_credential_name` value.

Parameter	Description
regex_filter	Specifies the REGEX expression to filter files. The REGEX expression pattern must be compatible with the REGEXP_LIKE operator. If you do not supply a regex_filter value, the regex_filter is set to NULL. See REGEXP_LIKE Condition for more information.
format	Specifies the additional configuration options for the file operation. These options are specified as a JSON string. The supported format options are: • logretention: It accepts an integer value that determines the duration in days for which the status table is retained for a bulk operation. The default value is 2 days. • logprefix: It accepts a string value that determines the bulk operation status table name prefix string. The operation type is the default value. For BULK_COPY, the default logprefix value is COPYOBJ. • priority: It accepts a string value that determines the number of file operations performed concurrently. An operation with a higher priority consumes more database resources and should run faster. It accepts the following values: – HIGH: Determines the number of parallel files handled using the database's **ECPU count** (**OCPU count** if your database uses OCPUs) – MEDIUM: Determines the number of simultaneous processes using the concurrency limit for Medium service. The default value is 4. – LOW: Process the files in serial order. The default value is MEDIUM. The maximum number of concurrent file operations is limited to 64. If you do not supply a format value, the format is set to NULL.
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Usage Notes

- An error is returned when the source and target URI point to the same Object Storage bucket or folder.

Example

```
BEGIN
DBMS_CLOUD.BULK_COPY (
     source_credential_name => 'OCI_CRED',
     source_location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/o',
     target_location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/o',
     format       => JSON_OBJECT ('logretention' value 7, 'logprefix' value
'BULKOP')
);
```

```
END;
/
```

BULK_DELETE Procedure

This procedure bulk deletes files from the Cloud Object Storage. The overloaded form enables you to use the `operation_id` parameter. You can filter the list of files to be deleted using a regular expression pattern compatible with `REGEXP_LIKE` operator.

Syntax

```
 DBMS_CLOUD.BULK_DELETE(
     credential_name  IN  VARCHAR2 DEFAULT NULL,
     location_uri     IN  VARCHAR2,
     regex_filter     IN  VARCHAR2 DEFAULT NULL,
     format           IN  CLOB     DEFAULT NULL
);

DBMS_CLOUD.BULK_DELETE (
     credential_name  IN  VARCHAR2 DEFAULT NULL,
     location_uri     IN  VARCHAR2,
     regex_filter     IN  VARCHAR2 DEFAULT NULL,
     format           IN  CLOB     DEFAULT NULL,
     operation_id     OUT NUMBER
);
```

Parameters

Parameter	Description
`credential_name`	The name of the credential to access the Cloud Object Storage. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. If you do not supply a `credential_name` value, the `credential_name` is set to `NULL`.
`location_uri`	Specifies URI, that point to an Object Storage location in the Autonomous Database. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD URI Formats for more information.
`regex_filter`	Specifies the REGEX expression to filter files. The REGEX expression pattern must be compatible with the `REGEXP_LIKE` operator. If you do not supply a `regex_filter` value, the `regex_filter` is set to `NULL`. See REGEXP_LIKE Condition for more information.

Parameter	Description
format	Specifies the additional configuration options for the file operation. These options are specified as a JSON string. The supported format options are: • logretention: It accepts an integer value that determines the duration in days for which the status table is retained for a bulk operation . The default value is 2 days. • logprefix: It accepts a string value that determines the bulk operation status table name prefix string. The operation type is the default value. For BULK_DELETE, the default logprefix value is DELETE. • priority: It accepts a string value that determines the number of file operations performed concurrently. An operation with a higher priority consumes more database resources and is completed sooner. It accepts the following values: – HIGH: Determines the number of parallel files handled using the database's **ECPU count** (**OCPU count** if your database uses OCPUs). – MEDIUM: Determines the number of simultaneous processes using the concurrency limit for Medium service. The default value is 4. – LOW: Process the files in serial order. The default value is MEDIUM. The maximum number of concurrent file operations is limited to 64. If you do not supply a format value, the format is set to NULL.
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Example

```
BEGIN
DBMS_CLOUD.BULK_DELETE (
     credential_name => 'OCI_CRED',
     location_uri    => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o',
     format          => JSON_OBJECT ('logretention' value 5, 'logprefix'
value 'BULKDEL')
);
END;
/
```

BULK_DOWNLOAD Procedure

This procedure downloads files into an Autonomous Database directory from Cloud Object Storage. The overloaded form enables you to use the operation_id parameter. You can filter the list of files to be downloaded using a regular expression pattern compatible with REGEXP_LIKE operator.

Syntax

```
DBMS_CLOUD.BULK_DOWNLOAD (
     credential_name  IN  VARCHAR2 DEFAULT NULL,
     location_uri     IN  VARCHAR2,
     directory_name   IN  VARCHAR2,
     regex_filter     IN  VARCHAR2 DEFAULT NULL,
     format           IN  CLOB     DEFAULT NULL
);

DBMS_CLOUD.BULK_DOWNLOAD (
     credential_name  IN  VARCHAR2 DEFAULT NULL,
     location_uri     IN  VARCHAR2,
     directory_name   IN  VARCHAR2,
     regex_filter     IN  VARCHAR2 DEFAULT NULL,
     format           IN  CLOB     DEFAULT NULL,
     operation_id     OUT NUMBER
);
```

Parameters

Parameter	Description
`credential_name`	The name of the credential to access the Cloud Object Storage. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. If you do not supply a `credential_name` value, the `credential_name` is set to `NULL`.
`location_uri`	Specifies URI, that point to an Object Storage location in the Autonomous Database. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD URI Formats for more information.
`directory_name`	The name of the directory on the Autonomous Database from where you want to download the files. This parameter is mandatory.
`regex_filter`	Specifies the REGEX expression to filter files. The REGEX expression pattern must be compatible with the `REGEXP_LIKE` operator. If you do not supply a `regex_filter` value, the `regex_filter` is set to `NULL`. See REGEXP_LIKE Condition for more information.

Parameter	Description
format	Specifies the additional configuration options for the file operation. These options are specified as a JSON string. The supported format options are: • logretention: It accepts an integer value that determines the duration in days for which the status table is retained for a bulk operation. The default value is 2 days. • logprefix: It accepts a string value that determines the bulk operation status table name prefix string. For BULK_DOWNLOAD, the default logprefix value is DOWNLOAD. The operation type is the default value. • priority: It accepts a string value that determines the number of file operations performed concurrently. An operation with a higher priority consumes more database resources and is completed sooner. It accepts the following values: – HIGH: Determines the number of parallel files handled using the database's **ECPU count** (**OCPU count** if your database uses OCPUs). – MEDIUM: Determines the number of simultaneous processes using the concurrency limit for Medium service. The default value is 4. – LOW: Process the files in serial order. The default value is MEDIUM. The maximum number of concurrent file operations is limited to 64. If you do not supply a format value, the format is set to NULL.
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Example

```
BEGIN
DBMS_CLOUD.BULK_DOWNLOAD (
     credential_name => 'OCI_CRED',
     location_uri    => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o',
     directory_name  => 'BULK_TEST',
     format          => JSON_OBJECT ('logretention' value 7, 'logprefix'
value 'BULKOP')
 );
END;
/
```

BULK_MOVE Procedure

This procedure bulk moves files from one Cloud Object Storage bucket or folder to another. The overloaded form enables you to use the operation_id parameter.

You can filter the list of files to be deleted using a regular expression pattern compatible with REGEXP_LIKE operator.

The source and target bucket or folder can be in the same or different Cloud Object store provider.

When the source and target are in distinct Object Stores or have different accounts with the same cloud provider, you can give separate credential names for the source and target locations.

The source credential name is by default also used by the target location when target credential name is not provided.

The first step in moving files is copying them to the target location, then deleting the source files, once they are successfully copied.

The object is renamed rather than moved if Object Store allows renaming operations between source and target locations.

Syntax

```
DBMS_CLOUD.BULK_MOVE (
      source_credential_name  IN  VARCHAR2 DEFAULT NULL,
      source_location_uri     IN  VARCHAR2,
      target_location_uri     IN  VARCHAR2,
      target_credential_name  IN  VARCHAR2 DEFAULT NULL,
      regex_filter            IN  VARCHAR2 DEFAULT NULL,
      format                  IN  CLOB     DEFAULT NULL
);

DBMS_CLOUD.BULK_MOVE (
      source_credential_name  IN  VARCHAR2 DEFAULT NULL,
      source_location_uri     IN  VARCHAR2,
      target_location_uri     IN  VARCHAR2,
      target_credential_name  IN  VARCHAR2 DEFAULT NULL,
      regex_filter            IN  VARCHAR2 DEFAULT NULL,
      format                  IN  CLOB     DEFAULT NULL,
      operation_id            OUT NUMBER
);
```

Parameters

Parameter	Description
`source_credential_name`	The name of the credential to access the source Cloud Object Storage. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. If you do not supply a `source_credential_name` value, the `credential_name` is set to `NULL`.
`source_location_uri`	Specifies URI, that point to the source Object Storage bucket or folder location. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD URI Formats for more information.

Parameter	Description
target_location_uri	Specifies the URI for the target Object Storage bucket or folder, where the files need to be moved. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD URI Formats for more information.
target_credential_name	The name of the credential to access the target Cloud Object Storage location. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. If you do not supply a target_credential_name value, the target_location_uri is set to the source_credential_name value.
regex_filter	Specifies the REGEX expression to filter files. The REGEX expression pattern must be compatible with the REGEXP_LIKE operator. If you do not supply a regex_filter value, the regex_filter is set to NULL. See REGEXP_LIKE Condition for more information.
format	Specifies the additional configuration options for the file operation. These options are specified as a JSON string. The supported format options are: • logretention: It accepts an integer value that determines the duration in days for which the status table is retained for a bulk operation. The default value is 2 days. • logprefix: It accepts a string value that determines the bulk operation status table name prefix string. The operation type is the default value. For BULK_MOVE, the default logprefix value is MOVE. • priority: It accepts a string value that determines the number of file operations performed concurrently. An operation with a higher priority consumes more database resources and is completed sooner. It accepts the following values: – HIGH: Determines the number of parallel files handled using the database's **ECPU count** (**OCPU count** if your database uses OCPUs). – MEDIUM: Determines the number of simultaneous processes using the concurrency limit for Medium service. The default value is 4. – LOW: Process the files in serial order. The default value is MEDIUM. The maximum number of concurrent file operations is limited to 64. If you do not supply a format value, the format is set to NULL.
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Example

```
BEGIN
DBMS_CLOUD.BULK_MOVE (
```

```
     source_credential_name => 'OCI_CRED',
     source_location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/o',
     target_location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/o',
     format                 => JSON_OBJECT ('logretention' value 7,
'logprefix' value 'BULKMOVE')
);
END;
/
```

> **Note:**
>
> An error is returned when the source and target URI point to the same Object Storage bucket or folder.

BULK_UPLOAD Procedure

This procedure copies files into Cloud Object Storage from an Autonomous Database directory. The overloaded form enables you to use the `operation_id` parameter.

Syntax

```
DBMS_CLOUD.BULK_UPLOAD (
     credential_name  IN  VARCHAR2 DEFAULT NULL,
     location_uri     IN  VARCHAR2,
     directory_name   IN  VARCHAR2,
     regex_filter     IN  VARCHAR2 DEFAULT NULL,
     format           IN  CLOB     DEFAULT NULL
);

DBMS_CLOUD.BULK_UPLOAD (
     credential_name  IN  VARCHAR2 DEFAULT NULL,
     location_uri     IN  VARCHAR2,
     directory_name   IN  VARCHAR2,
     regex_filter     IN  VARCHAR2 DEFAULT NULL,
     format           IN  CLOB     DEFAULT NULL,
     operation_id     OUT NUMBER
);
```

Parameters

Parameter	Description
`credential_name`	The name of the credential to access the Cloud Object Storage. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information. If you do not supply a `credential_name` value, the `credential_name` is set to `NULL`.

Parameter	Description
location_uri	Specifies URI, that points to an Object Storage location to upload files. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD URI Formats for more information.
directory_name	The name of the directory on the Autonomous Database from where you upload files. This parameter is mandatory.
regex_filter	Specifies the REGEX expression to filter files. The REGEX expression pattern must be compatible with REGEXP_LIKE operator. If you do not supply a regex_filter value, the regex_filter is set to NULL. See REGEXP_LIKE Condition for more information.
format	Specifies the additional configuration options for the file operation. These options are specified as a JSON string. The supported format options are: • logretention: It accepts an integer value that determines the duration in days for which the status table is retained for a bulk operation. The default value is 2 days. • logprefix: It accepts a string value that determines the bulk operation status table name prefix string. The operation type is the default value. For BULK_UPLOAD, the default logprefix value is UPLOAD. • priority: It accepts a string value that determines the number of file operations performed concurrently. An operation with a higher priority consumes more database resources and is completed sooner. It accepts the following values: – HIGH: Determines the number of parallel files handled using the database's **ECPU count** (**OCPU count** if your database uses OCPUs). – MEDIUM: Determines the number of simultaneous processes using the concurrency limit for Medium service. The default value is 4. – LOW: Process the files in serial order. The default value is MEDIUM. The maximum number of concurrent file operations is limited to 64. If you do not supply a format value, the format is set to NULL.
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Example

```
BEGIN
DBMS_CLOUD.BULK_UPLOAD (
     credential_name => 'OCI_CRED',
     location_uri    => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o',
     directory_name  => 'BULK_TEST',
     format          => JSON_OBJECT ('logretention' value 5, 'logprefix'
```

```
value 'BULKUPLOAD')
 );
END;
/
```

DBMS_CLOUD REST APIs

This section covers the `DBMS_CLOUD` REST APIs provided with Autonomous Database.

REST API	Description
GET_RESPONSE_HEADERS Function	This function returns the HTTP response headers as JSON data in a JSON object in Autonomous Database.
GET_RESPONSE_RAW Function	This function returns the HTTP response in RAW format Autonomous Database. This is useful if the HTTP response is expected to be binary format.
GET_RESPONSE_STATUS_CODE Function	This function returns the HTTP response status code as an integer in Autonomous Database. The status code helps to identify if the request is successful.
GET_RESPONSE_TEXT Function	This function returns the HTTP response in TEXT format (`VARCHAR2` or `CLOB`) in Autonomous Database. Usually, most Cloud REST APIs return JSON response in text format. This function is useful if you expect the HTTP response is in text format.
GET_API_RESULT_CACHE_SIZE Function	This function returns the configured result cache size.
SEND_REQUEST Function and Procedure	This function begins an HTTP request, gets the response, and ends the response in Autonomous Database. This function provides a workflow for sending a Cloud REST API request with arguments and a return response code and payload.
SET_API_RESULT_CACHE_SIZE Procedure	This procedure sets the maximum cache size for current session.

- DBMS_CLOUD REST API Overview
  When you use PL/SQL in your application and you need to call Cloud REST APIs you can use `DBMS_CLOUD.SEND_REQUEST` to send the REST API requests.
- DBMS_CLOUD REST API Constants
  Describes the `DBMS_CLOUD` constants for making `HTTP` requests using `DBMS_CLOUD.SEND_REQUEST`.
- DBMS_CLOUD REST API Results Cache
  You can save `DBMS_CLOUD` REST API results when you set the `cache` parameter to true with `DBMS_CLOUD.SEND_REQUEST`. The `SESSION_CLOUD_API_RESULTS` view describes the columns you can use when REST API results are saved.
- GET_RESPONSE_HEADERS Function
  This function returns the HTTP response headers as JSON data in a JSON object.
- GET_RESPONSE_RAW Function
  This function returns the HTTP response in RAW format. This is useful if the HTTP response is expected to be binary format.
- GET_RESPONSE_STATUS_CODE Function
  This function returns the HTTP response status code as an integer. The status code helps to identify if the request is successful.

- GET_RESPONSE_TEXT Function
  This function returns the HTTP response in TEXT format (VARCHAR2 or CLOB). Usually, most Cloud REST APIs return JSON response in text format. This function is useful if you expect the HTTP response is in text format.
- GET_API_RESULT_CACHE_SIZE Function
  This function returns the configured result cache size. The cache size value only applies for the current session.
- SEND_REQUEST Function and Procedure
  This function and procedure begins an HTTP request, gets the response, and ends the response. This function provides a workflow for sending a cloud REST API request with arguments and the function returns a response code and payload. If you use the procedure, you can view results and response details from the saved results with the SESSION_CLOUD_API_RESULTS view.
- SET_API_RESULT_CACHE_SIZE Procedure
  This procedure sets the maximum cache size for current session. The cache size value only applies for the current session.
- DBMS_CLOUD REST API Examples
  Shows examples using DBMS_CLOUD.SEND_REQUEST to create and delete an Oracle Cloud Infrastructure Object Storage bucket, and an example to list all compartments in the tenancy.

DBMS_CLOUD REST API Overview

When you use PL/SQL in your application and you need to call Cloud REST APIs you can use DBMS_CLOUD.SEND_REQUEST to send the REST API requests.

The DBMS_CLOUD REST API functions allow you to make HTTP requests using DBMS_CLOUD.SEND_REQUEST and obtain and save results. These functions provide a generic API that lets you call any REST API with the following supported cloud services:

- Oracle Cloud Infrastructure
  See API Reference and Endpoints for information on Oracle Cloud Infrastructure REST APIs.
- Amazon Web Services (AWS)
  See Guides and API References for information on Amazon Web Services REST APIs.
- Azure Cloud [1]
  See Azure REST API Reference for information on Azure REST APIs.
- Oracle Cloud Infrastructure Classic
  See All REST Endpoints for information on Oracle Cloud Infrastructure Classic REST APIs.
- GitHub Repository
  See GitHub REST API for more information.

DBMS_CLOUD REST API Constants

Describes the DBMS_CLOUD constants for making HTTP requests using DBMS_CLOUD.SEND_REQUEST.

DBMS_CLOUD supports GET, PUT, POST, HEAD and DELETE HTTP methods. The REST API method to be used for an HTTP request is typically documented in the Cloud REST API documentation.

[1] Support for Azure Cloud REST API calls is limited to the domain "blob.windows.net".

Name	Type	Value
METHOD_DELETE	VARCHAR2(6)	'DELETE'
METHOD_GET	VARCHAR2(3)	'GET'
METHOD_HEAD	VARCHAR2(4)	'HEAD'
METHOD_POST	VARCHAR2(4)	'POST'
METHOD_PUT	VARCHAR2(3)	'PUT'

DBMS_CLOUD REST API Results Cache

You can save DBMS_CLOUD REST API results when you set the cache parameter to true with DBMS_CLOUD.SEND_REQUEST. The SESSION_CLOUD_API_RESULTS view describes the columns you can use when REST API results are saved.

By default DBMS_CLOUD REST API calls do not save results for your session. In this case you use the DBMS_CLOUD.SEND_REQUEST function to return results.

When you use DBMS_CLOUD.SEND_REQUEST and set the cache parameter to TRUE, results are saved and you can view past results in the SESSION_CLOUD_API_RESULTS view. Saving and querying historical results of DBMS_CLOUD REST API requests can help you when you need to work with your previous results in your applications.

For example, to query recent DBMS_CLOUD REST API results, use the view SESSION_CLOUD_API_RESULTS:

```
SELECT timestamp FROM SESSION_CLOUD_API_RESULTS;
```

When you save DBMS_CLOUD REST API results with DBMS_CLOUD.SEND_REQUEST the saved data is only available within the same session (connection). After the session exits, the saved data is no longer available.

Use DBMS_CLOUD.GET_API_RESULT_CACHE_SIZE and DBMS_CLOUD.SET_API_RESULT_CACHE_SIZE to view and set the DBMS_CLOUD REST API cache size, and to disable caching.

- DBMS_CLOUD REST API Results cache_scope Parameter
  When you save DBMS_CLOUD REST API results with DBMS_CLOUD.SEND_REQUEST, access to the results in SESSION_CLOUD_API_RESULTS is provided based on the value of cache_scope.
- DBMS_CLOUD REST API SESSION_CLOUD_API_RESULTS View
  You can save DBMS_CLOUD REST API results when you set the cache parameter to true with DBMS_CLOUD.SEND_REQUEST. The SESSION_CLOUD_API_RESULTS view describes the columns you can use when REST API results are saved.

DBMS_CLOUD REST API Results cache_scope Parameter

When you save DBMS_CLOUD REST API results with DBMS_CLOUD.SEND_REQUEST, access to the results in SESSION_CLOUD_API_RESULTS is provided based on the value of cache_scope.

By default cache_scope is 'PRIVATE' and only the current user of the session can access the results. If you set the cache_scope to 'PUBLIC', then all session users can access the results. The default value for cache_scope specifies that each user can only see DBMS_CLOUD.SEND_REQUEST REST API results generated by the procedures they invoke with invoker's rights. When you invoke DBMS_CLOUD.SEND_REQUEST in a session, there are three possibilities that determines if the current user can see results in the cache, based on the cache_scope value:

- You directly execute `DBMS_CLOUD.SEND_REQUEST` as a top-level statement and the call to `DBMS_CLOUD.SEND_REQUEST` and the REST API results are saved with the same username. In this case you have access to all results with the default value, `'PRIVATE'`, set for `cache_scope`.
- You write a wrapper invoker's rights procedure and as the current user your call with `DBMS_CLOUD.SEND_REQUEST` calls the procedure and the REST API results are saved with the same username. In this case, and you have access to all results with the default value, `'PRIVATE'`, set for `cache_scope`.
- You write a wrapper definer's rights procedure and the procedure is owned by another user. When you call `DBMS_CLOUD.SEND_REQUEST` inside the procedure, the results are saved with the username of the procedure owner.

  For this case, a different definer's rights user is invoking `DBMS_CLOUD.SEND_REQUEST`, and the REST API results are saved with that definers procedure's owner. For this case, by default when `cache_scope` is `PRIVATE'`, the invoker's session cannot see the results.

  If the definer's procedure owner wants to make the results available to any invoking session user, then they must set `cache_scope` to `'PUBLIC'` in the `DBMS_CLOUD.SEND_REQUEST`.

DBMS_CLOUD REST API SESSION_CLOUD_API_RESULTS View

You can save `DBMS_CLOUD` REST API results when you set the `cache` parameter to true with `DBMS_CLOUD.SEND_REQUEST`. The `SESSION_CLOUD_API_RESULTS` view describes the columns you can use when REST API results are saved.

The view `SESSION_CLOUD_API_RESULTS` is the view created if you cache results with `DBMS_CLOUD.SEND_REQUEST`. You can query historical results which belong to your user session. When the session ends, the data in the `SESSION_CLOUD_API_RESULTS` is purged.

Column	Description
`URI`	The `DBMS_CLOUD` REST API request URL
`TIMESTAMP`	The `DBMS_CLOUD` REST API response timestamp
`CLOUD_TYPE`	The `DBMS_CLOUD` REST API cloud type, such as Oracle Cloud Infrastructure, AMAZON_S3, and AZURE_BLOB
`REQUEST_METHOD`	The `DBMS_CLOUD` REST API request method, such as `GET`, `PUT`, `HEAD`
`REQUEST_HEADERS`	The `DBMS_CLOUD` REST API request headers
`REQUEST_BODY_TEXT`	The `DBMS_CLOUD` REST API request body in `CLOB`
`RESPONSE_STATUS_CODE`	The `DBMS_CLOUD` REST API response status code, such as `200(OK)`, `404(Not Found)`
`RESPONSE_HEADERS`	The `DBMS_CLOUD` REST API response headers
`RESPONSE_BODY_TEXT`	The `DBMS_CLOUD` REST API response body in `CLOB`
`SCOPE`	The `cache_scope` set by `DBMS_CLOUD.SEND_REQUEST`. Valid values are `PUBLIC` or `PRIVATE`.

GET_RESPONSE_HEADERS Function

This function returns the HTTP response headers as JSON data in a JSON object.

Syntax

```
DBMS_CLOUD.GET_RESPONSE_HEADERS(
       resp           IN DBMS_CLOUD_TYPES.resp)
   RETURN JSON_OBJECT_T;
```

Parameters

Parameter	Description
resp	HTTP Response type returned from DBMS_CLOUD.SEND_REQUEST.

Exceptions

Exception	Error	Description
invalid_response	ORA-20025	Invalid response type object passed to DBMS_CLOUD.GET_RESPONSE_HEADERS.

GET_RESPONSE_RAW Function

This function returns the HTTP response in RAW format. This is useful if the HTTP response is expected to be binary format.

Syntax

```
DBMS_CLOUD.GET_RESPONSE_RAW(
       resp           IN DBMS_CLOUD_TYPES.resp)
   RETURN BLOB;
```

Parameters

Parameter	Description
resp	HTTP Response type returned from DBMS_CLOUD.SEND_REQUEST.

Exceptions

Exception	Error	Description
invalid_response	ORA-20025	Invalid response type object passed to DBMS_CLOUD.GET_RESPONSE_RAW.

GET_RESPONSE_STATUS_CODE Function

This function returns the HTTP response status code as an integer. The status code helps to identify if the request is successful.

Syntax

```
DBMS_CLOUD.GET_RESPONSE_STATUS_CODE(
       resp           IN DBMS_CLOUD_TYPES.resp)
   RETURN PLS_INTEGER;
```

Parameters

Parameter	Description
resp	HTTP Response type returned from DBMS_CLOUD.SEND_REQUEST.

Exceptions

Exception	Error	Description
invalid_response	ORA-20025	Invalid response type object passed to DBMS_CLOUD.GET_RESPONSE_STATUS_CODE.

GET_RESPONSE_TEXT Function

This function returns the HTTP response in TEXT format (VARCHAR2 or CLOB). Usually, most Cloud REST APIs return JSON response in text format. This function is useful if you expect the HTTP response is in text format.

Syntax

```
DBMS_CLOUD.GET_RESPONSE_TEXT(
       resp           IN DBMS_CLOUD_TYPES.resp)
   RETURN CLOB;
```

Parameters

Parameter	Description
resp	HTTP Response type returned from DBMS_CLOUD.SEND_REQUEST.

Exceptions

Exception	Error	Description
invalid_response	ORA-20025	Invalid response type object passed to DBMS_CLOUD.GET_RESPONSE_TEXT.

GET_API_RESULT_CACHE_SIZE Function

This function returns the configured result cache size. The cache size value only applies for the current session.

Syntax

```
DBMS_CLOUD.GET_API_RESULT_CACHE_SIZE()
   RETURN NUMBER;
```

SEND_REQUEST Function and Procedure

This function and procedure begins an HTTP request, gets the response, and ends the response. This function provides a workflow for sending a cloud REST API request with arguments and the function returns a response code and payload. If you use the procedure, you can view results and response details from the saved results with the `SESSION_CLOUD_API_RESULTS` view.

Syntax

```
DBMS_CLOUD.SEND_REQUEST(
       credential_name    IN VARCHAR2,
       uri                IN VARCHAR2,
       method             IN VARCHAR2,
       headers            IN CLOB DEFAULT NULL,
       async_request_url  IN VARCHAR2 DEFAULT NULL,
       wait_for_states    IN DBMS_CLOUD_TYPES.wait_for_states_t DEFAULT NULL,
       timeout            IN NUMBER DEFAULT 0,
       cache              IN PL/SQL BOOLEAN DEFAULT FALSE,
       cache_scope        IN VARCHAR2 DEFAULT 'PRIVATE',
       body               IN BLOB DEFAULT NULL)
   RETURN DBMS_CLOUD_TYPES.resp;

DBMS_CLOUD.SEND_REQUEST(
       credential_name    IN VARCHAR2,
       uri                IN VARCHAR2,
       method             IN VARCHAR2,
       headers            IN CLOB DEFAULT NULL,
       async_request_url  IN VARCHAR2 DEFAULT NULL,
       wait_for_states    IN DBMS_CLOUD_TYPES.wait_for_states_t DEFAULT NULL,
       timeout            IN NUMBER DEFAULT 0,
       cache              IN PL/SQL BOOLEAN DEFAULT FALSE,
       cache_scope        IN VARCHAR2 DEFAULT 'PRIVATE',
       body               IN BLOB DEFAULT NULL);
```

Parameters

Parameter	Description
`credential_name`	The name of the credential for authenticating with the corresponding cloud native API. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL for more information.
`uri`	HTTP URI to make the request.
`method`	HTTP Request Method: `GET`, `PUT`, `POST`, `HEAD`, `DELETE`. Use the `DBMS_CLOUD` package constant to specify the method. See DBMS_CLOUD REST API Constants for more information.

Parameter	Description
headers	HTTP Request headers for the corresponding cloud native API in JSON format. The authentication headers are set automatically, only pass custom headers.
async_request_url	An asynchronous request URL. To obtain the URL select your request API from the list of APIs (see https://docs.cloud.oracle.com/en-us/iaas/api/). Then, navigate to find the API for your request in the left pane. For example, Database Services API → Autonomous Database → StopAutonomousDatabase. This page shows the API home (and shows the base endpoint). Then, append the base endpoint with the relative path obtained for your work request WorkRequest link.
wait_for_states	Wait for states is a status of type: DBMS_CLOUD_TYPES.wait_for_states_t. The following are valid values for expected states: 'ACTIVE', 'CANCELED', 'COMPLETED', 'DELETED', 'FAILED', 'SUCCEEDED'. Multiple states are allowed for wait_for_states. The default value for wait_for_states is to wait for any of the expected states: 'ACTIVE', 'CANCELED', 'COMPLETED', 'DELETED', 'FAILED', 'SUCCEEDED'.
timeout	Specifies the timeout, in seconds, for asynchronous requests with the parameters async_request_url and wait_for_states. Default value is 0. This indicates to wait for completion of the request without any timeout.
cache	If TRUE specifies the request should be cached in REST result API cache. The default value is FALSE, which means REST API requests are not cached.
cache_scope	Specifies whether everyone can have access to this request result cache. Valid values: "PRIVATE" and "PUBLIC". The default value is "PRIVATE".
body	HTTP Request Body for PUT and POST requests.

Exceptions

Exception	Error	Description
invalid_req_method	ORA-20023	Request method passed to DBMS_CLOUD.SEND_REQUEST is invalid.
invalid_req_header	ORA-20024	Request headers passed to DBMS_CLOUD.SEND_REQUEST are not in valid JSON format.

Usage Notes

- If you are using Oracle Cloud Infrastructure, you must use a Signing Key based credential value for the credential_name. See CREATE_CREDENTIAL Procedure for more information.
- The optional parameters async_request_url, wait_for_states, and timeout allow you to handle long running requests. Using this asynchronous form of send_request, the function waits for the completion status specified in wait_for_states before returning. With these parameters in the send request, you pass the expected return states in the wait_for_states parameter, and you use the async_request_url parameter to specify an

associated work request, the request does not return immediately. Instead, the request probes the `async_request_url` until the return state is one of the expected states or the `timeout` is exceeded (`timeout` is optional). If no `timeout` is specified, the request waits until a state found in `wait_for_states` occurs.

SET_API_RESULT_CACHE_SIZE Procedure

This procedure sets the maximum cache size for current session. The cache size value only applies for the current session.

Syntax

```
DBMS_CLOUD.SET_API_RESULT_CACHE_SIZE(
       cache_size          IN NUMBER);
```

Parameters

Parameter	Description
`cache_size`	Set the maximum cache size to the specified value `cache_size`. If the new maximum cache size is smaller than the current cache size, older records are dropped until the number of rows is equal to the specified maximum cache size. The maximum value is `10000`. If the cache size is set to `0`, caching is disabled in the session. The default cache size is `10`.

Exceptions

Exception	Error	Description
`invalid API result cache size`	`ORA-20032`	The minimum value is 0 and the maximum value is 10000. This exception is shown when the input value is less than 0 or is larger than 10000.

Example

```
EXEC DBMS_CLOUD.SET_API_RESULT_CACHE_SIZE(101);
```

DBMS_CLOUD REST API Examples

Shows examples using `DBMS_CLOUD.SEND_REQUEST` to create and delete an Oracle Cloud Infrastructure Object Storage bucket, and an example to list all compartments in the tenancy.

> **Note:**
>
> These examples show Oracle Cloud Infrastructure request APIs and require that you use a Signing Key based credential for the `credential_name`. Oracle Cloud Infrastructure Signing Key based credentials include the `private_key` and `fingerprint` arguments.
>
> For example:
>
> ```
> BEGIN
>    DBMS_CLOUD.CREATE_CREDENTIAL (
>        credential_name => 'OCI_KEY_CRED',
>        user_ocid       =>
> 'ocid1.user.oc1..aaaaaaaauq54mi7zdyfhw33ozkwuontjceel7fok5nq3bf2vwetkp
> qsoa',
>        tenancy_ocid    =>
> 'ocid1.tenancy.oc1..aabbbbbbaafcue47pqmrf4vigneebgbcmmoy5r7xvoypicjqqg
> e32ewnrcyx2a',
>        private_key     =>
> 'MIIEogIBAAKCAQEAtUnxbmrekwgVac6FdWeRzoXvIpA9+0r1.....wtnNpESQQQ0QLGPD
> 8NM//JEBg=',
>        fingerprint     =>
> 'f2:db:f9:18:a4:aa:fc:94:f4:f6:6c:39:96:16:aa:27');
> END;
> /
> ```
>
> See CREATE_CREDENTIAL Procedure for information on `DBMS_CLOUD.CREATE_CREDENTIAL`.

Create Bucket Example

Shows an example using `DBMS_CLOUD.SEND_REQUEST` with HTTP `POST` method to create an object store bucket named *`bucketname`*.

See CreateBucket for details on the Oracle Cloud Infrastructure Object Storage Service API for this example.

```
SET SERVEROUTPUT ON
DECLARE
  resp DBMS_CLOUD_TYPES.resp;
BEGIN
  -- Send request
  resp := DBMS_CLOUD.send_request(
            credential_name => 'OCI_KEY_CRED',
            uri => 'https://objectstorage.region.oraclecloud.com/n/namespace-
string/b/',
            method => DBMS_CLOUD.METHOD_POST,
            body => UTL_RAW.cast_to_raw(
                        JSON_OBJECT('name' value 'bucketname',
                                    'compartmentId' value 'compartment_OCID'))
          );

  -- Response Body in TEXT format
  dbms_output.put_line('Body: ' || '------------' || CHR(10) ||
```

```
  DBMS_CLOUD.get_response_text(resp) || CHR(10));

  -- Response Headers in JSON format
  dbms_output.put_line('Headers: ' || CHR(10) || '------------' || CHR(10) ||
  DBMS_CLOUD.get_response_headers(resp).to_clob || CHR(10));

  -- Response Status Code
  dbms_output.put_line('Status Code: ' || CHR(10) || '------------' ||
CHR(10) ||
  DBMS_CLOUD.get_response_status_code(resp));

END;
/
```

Notes:

- In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.
- Where: *region* is an endpoint region. See Object Storage API reference in API Reference and Endpoints for more information. For example, where *region* is: `us-phoenix-1`.

Delete Bucket Example

Shows an example using `DBMS_CLOUD.SEND_REQUEST` with HTTP `DELETE` method to delete an object store bucket named *bucketname*.

See DeleteBucket for details on the Oracle Cloud Infrastructure Object Storage Service API for this example.

```
SET SERVEROUTPUT ON
DECLARE
  resp DBMS_CLOUD_TYPES.resp;
BEGIN
  -- Send request
  resp := DBMS_CLOUD.send_request(
            credential_name => 'OCI_KEY_CRED',
            uri => 'https://objectstorage.region.oraclecloud.com/n/namespace-
string/b/bucketname',
            method => DBMS_CLOUD.METHOD_DELETE
          );

  -- Response Body in TEXT format
  dbms_output.put_line('Body: ' || '------------' || CHR(10) ||
  DBMS_CLOUD.get_response_text(resp) || CHR(10));

  -- Response Headers in JSON format
  dbms_output.put_line('Headers: ' || CHR(10) || '------------' || CHR(10) ||
  DBMS_CLOUD.get_response_headers(resp).to_clob || CHR(10));

  -- Response Status Code
  dbms_output.put_line('Status Code: ' || CHR(10) || '------------' ||
CHR(10) ||
  DBMS_CLOUD.get_response_status_code(resp));
```

```
END;
/
```

Notes:

- In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.
- Where: *region* is an endpoint region. See Object Storage API reference in API Reference and Endpoints for more information. For example, where *region* is: `us-phoenix-1`.

List Compartments Example

Shows an example using `DBMS_CLOUD.SEND_REQUEST` with HTTP `GET` method to list all compartments in the tenancy (root compartment). This example shows how to pass request headers in the `DBMS_CLOUD.SEND_REQUEST`.

See ListCompartments for details on the Oracle Cloud Infrastructure Identity and Access Management Service API for this example.

```
--
-- List compartments
--
DECLARE
  resp DBMS_CLOUD_TYPES.resp;
  root_compartment_ocid VARCHAR2(512) := '&1';
BEGIN
  -- Send request
  dbms_output.put_line('Send Request');
  resp := DBMS_CLOUD.send_request(
            credential_name => 'OCI_KEY_CRED',
            uri => 'https://identity.region.oraclecloud.com/20160918/
compartments?compartmentId=' || root_compartment_ocid,
            method => DBMS_CLOUD.METHOD_GET,
            headers => JSON_OBJECT('opc-request-id' value 'list-compartments')
          );
  dbms_output.put_line('Body: ' || '------------' || CHR(10) ||
DBMS_CLOUD.get_response_text(resp) || CHR(10));
  dbms_output.put_line('Headers: ' || CHR(10) || '------------' || CHR(10) ||
DBMS_CLOUD.get_response_headers(resp).to_clob || CHR(10));
  dbms_output.put_line('Status Code: ' || CHR(10) || '------------' ||
CHR(10) || DBMS_CLOUD.get_response_status_code(resp));
  dbms_output.put_line(CHR(10));
END;
/
```

Where: *region* is an endpoint region. See Identity and Access Management (IAM) API reference in API Reference and Endpoints for more information. For example, where *region* is: `uk-london-1`.

Asynchronous Request Example

Shows an example using `DBMS_CLOUD.SEND_REQUEST` with HTTP `POST` method to perform the Autonomous Database stop operation and wait for status. This example shows how to use

DBMS_CLOUD.SEND_REQUEST with the async_request_url, wait_for_states, and timeout parameters.

```
--
-- Sent Work Request Autonomous Database Stop Request with Wait for Status
DECLARE
    l_resp DBMS_CLOUD_TYPES.resp;
    l_resp_json JSON_OBJECT_T;
    l_key_shape JSON_OBJECT_T;
    l_body JSON_OBJECT_T;
    status_array DBMS_CLOUD_TYPES.wait_for_states_t;
BEGIN
  status_array := DBMS_CLOUD_TYPES.wait_for_states_t('SUCCEEDED');
  l_body := JSON_OBJECT_T('{}');
  l_body.put('autonomousDatabaseId', 'ocid');
-- Send request
  dbms_output.put_line(l_body.to_clob);
  dbms_output.put_line('Send Request');
  l_resp := DBMS_CLOUD.send_request(
                        credential_name    => 'NATIVE_CRED_OCI',
                        uri                => 'https://
database.region.oraclecloud.com/20160918/autonomousDatabases/ocid/actions/
stop',
                        method             => DBMS_CLOUD.METHOD_POST,
                        body               =>
UTL_RAW.cast_to_raw(l_body.to_clob),
                        async_request_url  => 'https://
iaas.region.oraclecloud.com/20160918/workRequests',
                        wait_for_states    => status_array,
                        timeout            => 600
                    );
   dbms_output.put_line('resp body: '||DBMS_CLOUD.get_response_text(l_resp));
   dbms_output.put_line('resp headers: '||
DBMS_CLOUD.get_response_headers(l_resp).to_clob);
END;
/
```

Where: *region* is an endpoint region. See Identity and Access Management (IAM) API reference in API Reference and Endpoints for more information. For example, where *region* is: uk-london-1.

The *ocid* is the Oracle Cloud Infrastructure resource identifier. See Resource Identifiers for more information.

DBMS_CLOUD URI Formats

Describes the format of the source file URIs in operations with DBMS_CLOUD. The format depends on the object storage service you are using.

DBMS_CLOUD guarantees secure communication and any URI that you specify must use HTTPS, with https:// as the prefix for the URI.

- Oracle Cloud Infrastructure Object Storage Native URI Format
- Oracle Cloud Infrastructure Object Storage Swift URI Format

- Oracle Cloud Infrastructure Object Storage URI Format Using Pre-Authenticated Request URL
- URI Format Using Public URL
- Oracle Cloud Infrastructure Object Storage Classic URI Format
- Amazon S3 URI Format
- Azure Blob Storage URI Format
- Amazon S3 Compatible URI Format
- GitHub Raw URL Format
  `DBMS_CLOUD` supports GitHub Raw URLs to access data from a GitHub Repository.
- Additional Customer-Managed URI Formats

Oracle Cloud Infrastructure Object Storage Native URI Format

If your source files reside on the Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure native URIs, with the format:

```
https://objectstorage.region.oraclecloud.com/n/namespace-string/b/bucket/o/
filename
```

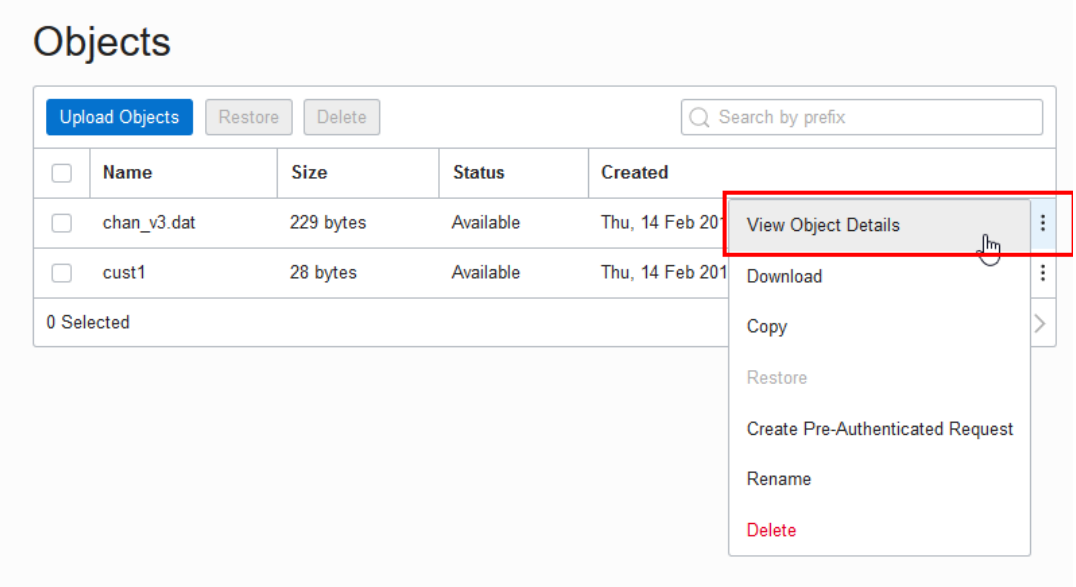
For example, the Native URI for the file `channels.txt` in the *`bucketname`* bucket in the `Phoenix` data center is:

```
https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o/channels.txt
```

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

You can find the URI from the Oracle Cloud Infrastructure Object Storage "Object Details" in the right hand side ellipsis menu in the Object Store:

1. Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
2. From the Oracle Cloud Infrastructure left navigation menu click **Core Infrastructure**. Under **Object Storage**, click **Object Storage**.
3. Under List Scope, select a **Compartment**.
4. From the **Name** column, select a bucket.
5. In the Objects area, click **View Object Details**.



6. On the **Object Details** page, the **URL Path (URI)** field shows the URI to access the object.

> **Note:**
>
> The source files need to be stored in an Object Storage tier bucket. Autonomous Database does not support buckets in the Archive Storage tier. See Overview of Object Storage for more information.

Oracle Cloud Infrastructure Object Storage Swift URI Format

If your source files reside on the Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure Swift URIs with the format:

```
https://swiftobjectstorage.region.oraclecloud.com/v1/namespace-string/bucket/
filename
```

For example, the Swift URI for the file `channels.txt` in the *`bucketname`* bucket in the `Phoenix` data center is:

```
https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/namespace-string/
bucketname/channels.txt
```

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

> **Note:**
>
> The source files need to be stored in an Object Storage tier bucket. Autonomous Database does not support buckets in the Archive Storage tier. See Overview of Object Storage for more information.

Oracle Cloud Infrastructure Object Storage URI Format Using Pre-Authenticated Request URL

If your source files reside on the Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure pre-authenticated URIs. When you create a pre-authenticated request, a unique URL is generated. You can then provide the unique URL to users in your organization, partners, or third parties to access the Object Storage resource target identified in the pre-authenticated request.

> **Note:**
>
> Carefully assess the business requirement for and the security ramifications of pre-authenticated access. When you create the pre-authenticated request URL, note the **Expiration** and the **Access Type** to make sure they are appropriate for your use. A pre-authenticated request URL gives anyone who has the URL access to the targets identified in the request for as long as the request is active. In addition to considering the operational needs of pre-authenticated access, it is equally important to manage its distribution.

The format for pre-authenticated request URLs is:

`https://objectstorage.`*`region`*`.oraclecloud.com/p/`*`encrypted_string`*`/n/`*`namespace-string`*`/b/`*`bucket`*`/o/`*`filename`*

For example, a sample pre-authenticated URI for the file `channels.txt` in the *bucketname* bucket in the `Phoenix` data center is:

`https://objectstorage.us-phoenix-1.oraclecloud.com/p/2xN-uDtWJNsiD910UCYGue/n/`*`namespace-string`*`/b/`*`bucketname`*`/o/channels.txt`

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

You can use a pre-authenticated URL in any `DBMS_CLOUD` procedure that takes a URL to access files in Oracle Cloud Infrastructure object store, without the need to create a credential. You need to either specify the `credential_name` parameter as `NULL` or not supply a `credential_name` parameter.

For example:

```
BEGIN
  DBMS_CLOUD.COPY_DATA(
     table_name =>'CHANNELS',
     file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/p/
unique-pre-authenticated-string/n/namespace-string/b/bucketname/o/
channels.txt',
```

```
     format => json_object('delimiter' value ',') );
END;
/
```

> **Note:**
>
> A list of mixed URLs is valid. If the URL list contains both pre-authenticated URLs and URLs that require authentication, `DBMS_CLOUD` uses the specified `credential_name` to access the URLs that require authentication and for the pre-authenticated URLs the specified `credential_name` is ignored.

See Using Pre-Authenticated Requests for more information.

URI Format Using Public URL

If your source files reside on an Object Store that provides public URLs, you can use public URLs with `DBMS_CLOUD` procedures. Public means the Object Storage service supports anonymous, unauthenticated access to the Object Store files. See your Cloud Object Storage service for details on how to make an object public in a supported Object Store.

> **Note:**
>
> Carefully assess the business requirement for and the security ramifications of using public URLs. When you use public URLs, due to the file content not being authenticated, make sure this is appropriate for your use.

You can use a public URL in any `DBMS_CLOUD` procedure that takes a URL to access files in your object store, without the need to create a credential. You need to either specify the `credential_name` parameter as `NULL` or not supply a `credential_name` parameter.

For example the following uses `DBMS_CLOUD.COPY_DATA` without a `credential_name`:

```
BEGIN
  DBMS_CLOUD.COPY_DATA(
     table_name =>'CHANNELS',
     file_uri_list =>'https://objectstorage.us-ashburn-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/chan_v3.dat',
     format => json_object('delimiter' value ',') );
END;
/
```

In this example, *`namespace-string`* is the Oracle Cloud Infrastructure object storage namespace and *`bucketname`* is the bucket name. See Understanding Object Storage Namespaces for more information.

> **Note:**
>
> A list of mixed URLs is valid. If the URL list contains both public URLs and URLs that require authentication, `DBMS_CLOUD` uses the specified `credential_name` to access the URLs that require authentication and for the public URLs the specified `credential_name` is ignored.

See Public Buckets for information on using Oracle Cloud Infrastructure public buckets.

Oracle Cloud Infrastructure Object Storage Classic URI Format

If your source files reside in Oracle Cloud Infrastructure Object Storage Classic, see the REST page for a description of the URI format for accessing your files: About REST URLs for Oracle Cloud Infrastructure Object Storage Classic Resources.

Amazon S3 URI Format

If your source files reside in Amazon S3, see the following for a description of the URI format for accessing your files: Accessing a bucket.

For example the following refers to the file `channels.txt` in the `adb` bucket in the `us-west-2` region.

```
https://s3-us-west-2.amazonaws.com/adb/channels.txt
```

You can use a presigned URL in any `DBMS_CLOUD` procedure that takes a URL to access files in Amazon S3 object store, without the need to create a credential. To use a presigned URL in any `DBMS_CLOUD` procedure, either specify the `credential_name` parameter as `NULL`, or do not supply a `credential_name` parameter.

See Share an Object with Others for more information.

> **Note:**
>
> `DBMS_CLOUD` supports the standard Amazon S3 endpoint syntax to access your buckets. `DBMS_CLOUD` does not support Amazon S3 legacy endpoints. See Legacy Endpoints for more information.

Azure Blob Storage URI Format

If your source files reside in Azure Blob Storage, see the following for a description of the URI format for accessing your files: Resource URI Syntax.

For example the following refers to the file `channels.txt` in the `adb` container in the storage account `adb_user`:

```
https://adb_user.blob.core.windows.net/adb/channels.txt
```

> **Note:**
>
> You can use Shared Access Signatures (SAS) URL in any `DBMS_CLOUD` procedure that takes a URL to access files in Azure Blob Storage, without the need to create a credential. To use a Shared Access Signature (SAS) URL, either specify the `credential_name` parameter as `NULL`, or do not supply a `credential_name` parameter.
> See Grant Limited Access to Azure Storage Resources Using Shared Access Signatures (SAS) for more information.

Amazon S3 Compatible URI Format

`DBMS_CLOUD` supports object storage service implementations that support Amazon S3 compatible URLs, including the following services:

- Oracle Cloud Infrastructure Object Storage with Amazon S3 compatible URL
- Google Cloud Storage with Amazon S3 compatible URL
- Wasabi Hot Cloud Storage with Amazon S3 compatible URL

> **Note:**
>
> To use `DBMS_CLOUD` with an Amazon S3 compatible object store you need to provide valid credentials. See CREATE_CREDENTIAL Procedure for more information.

If your source files reside on a service that supports Amazon S3 compatible URIs, use the following URI format to access your files:

- **Oracle Cloud Infrastructure Object Storage S3 Compatible URL**

  Object URL Format:

  ```
  https://mynamespace.compat.objectstorage.region.oraclecloud.com/
  bucket_name/object_name
  ```

  Bucket URL Format:

  ```
  https://mynamespace.compat.objectstorage.region.oraclecloud.com/bucket_name
  ```

  See Amazon S3 Compatibility and Object Storage Service API for more information.

- **Google Cloud Storage S3 Compatible URL**

  Object URL Format:

  ```
  https://bucketname.storage.googleapis.com/object_name
  ```

  Bucket URL Format:

  ```
  https://bucketname.storage.googleapis.com/
  ```

See Migrating from Amazon S3 to Cloud Storage and Request Endpoints for more information.

- **Wasabi S3 Compatible URL**

  Object URL Format:

  ```
  https://bucketname.s3.region.wasabisys.com/object_name
  ```

  Bucket URL Format:

  ```
  https://bucketname.s3.region.wasabisys.com/
  ```

  See S3-compatible API Connectivity for Wasabi Hot Cloud Storage and What are the service URLs for Wasabi's different regions? for more information.

GitHub Raw URL Format

`DBMS_CLOUD` supports GitHub Raw URLs to access data from a GitHub Repository.

> **Note:**
>
> For `DBMS_CLOUD` access with GitHub Raw URLs, repository access is limited to read-only functionality. The `DBMS_CLOUD` APIs such as `DBMS_CLOUD.PUT_OBJECT` that write data are not supported with `DBMS_CLOUD` APIs on a GitHub Repository.
> As an alternative, use `DBMS_CLOUD_REPO.PUT_FILE` to upload data to a GitHub Repository.

Use GitHub Raw URLs with `DBMS_CLOUD` APIs to access source files that reside on a GitHub Repository. When you browse to a file on GitHub and click the **Raw** link, this shows the GitHub Raw URL. The `raw.githubusercontent.com` domain provides unprocessed versions of files stored in GitHub repositories.

For example, using `DBMS_CLOUD.GET_OBJECT`:

```
BEGIN
 DBMS_CLOUD.GET_OBJECT(
   credential_name => 'MY_CRED',
   object_uri      => 'https://raw.githubusercontent.com/myaccount/myrepo/
master/data-management-library/autonomous-database/adb-loading.csv',
   directory_name  => 'DATA_PUMP_DIR'
  );
END;
/
```

For example, using `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`:

```
BEGIN
  DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
    credential_name => 'MY_CRED',
    table_name      => 'EMPLOYEES_EXT',
    file_uri_list   => 'https://raw.githubusercontent.com/myaccount/myrepo/
```

```
master/data-management-library/autonomous-database/*.csv',
    column_list     => 'name varchar2(30), gender varchar2(30), salary
number',
    format          => JSON_OBJECT('type' value 'csv')
  );
END;
/
SELECT * FROM employees_ext;
```

`DBMS_CLOUD` procedures that take a URL to access a GitHub Repository do not require credentials with public visibility GitHub repositories. To use a public visibility URL you can specify the `credential_name` parameter as `NULL` or not supply a `credential_name` parameter. See Setting repository visibility for more information.

Additional Customer-Managed URI Formats

In addition to the pre-configured, recognized `URIs` with their fully-qualified domain names (`FQDNs`), `DBMS_CLOUD` cannot determine the proper authentication scheme for customer-managed endpoints `URIs`. In those cases, `DBMS_CLOUD` relies on the proper `URI` scheme to identify the authentication scheme for the customer-managed endpoint.

URI Scheme	Authentication Type	Access Method Description	URI Example
basic://	Basic authentication	Username and password stored in database credential object is used to authenticate the HTTP request	`basic://api.github.com/users/myaccount`
bearer://	Bearer token authentication	Bearer token stored in the password field in database credential object is used to specify the Authorization header for the HTTP request	`bearer://api.sendgrid.com/v3/resource`
oci://	OCI native	OCI signing key obtained from database credential object stored and used to sign requests using the OCI authentication protocol	`oci://objectstorage.us-ashburn-1.oraclecloud.com`
public://	No authentication	Public URLs	`public://cms.data.gov/`
s3://	Amazon Web Services S3-compatible	Access key and secret key obtained from the username/password field of database credential object, and S3-compatible authentication performed for the HTTP request.	`s3://bucket.myprivatesite.com/file1.csv`

Examples:

Customer-managed endpoint using S3-compatible authentication.

This example shows how for new URIs, customers can add the public or private host name pattern using DBMS_NETWORK_ACL_ADMIN package. The code block, executed by user ADMIN, enables HTTPS access for user SCOTT to endpoints in domain *.myprivatesite.com. It then shows how user SCOTT accesses the newly enabled endpoint. Note that credential MY_CRED for user SCOTT must store the access key and secret key for S3-compatible authentication performed for the HTTP request indicated by the URI prefix.

```
BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => '*.myprivatesite.com',
         ace => xs$ace_type(privilege_list => xs$name_list('http'),
                            principal_name => 'SCOTT',
                            principal_type => xs_acl.ptype_db),
         private_target => TRUE   );
END;
/

BEGIN
   DBMS_CLOUD.get_object(
       credential_name => 'MY_CRED',
       object_uri      => 's3://bucket.myprivatesite.com/file1.csv',
       directory_name  => 'MY_DIR'  );
END;
/
```

Customer-managed endpoint with public access

This example shows how to register the SCOTT user to access public REST APIs. The ADMIN user creates a network ACL for the host to provide access to SCOTT user.

```
BEGIN
   DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'data.cms.gov',
         ace => xs$ace_type(privilege_list => xs$name_list('http'),
                            principal_name => 'SCOTT',
                            principal_type => xs_acl.ptype_db)
   );
END;
/

SELECT DBMS_CLOUD.get_response_text(
          DBMS_CLOUD.send_request(
              uri     => 'public://data.cms.gov/provider-data/api/1/datastore/
imports/a',
              method  => DBMS_CLOUD.METHOD_GET,
              headers => JSON_OBJECT('Accept' VALUE 'application/json')
          )
       )
   FROM DUAL;
/
```

DBMS_CLOUD Package Format Options

The format argument in `DBMS_CLOUD` specifies the format of source files.

The two ways to specify the format argument are:

```
format => '{"format_option" : "format_value" }'
```

And:

```
format => json_object('format_option' value 'format_value'))
```

Examples:

```
format => json_object('type' VALUE 'CSV')
```

To specify multiple format options, separate the values with a ",".

For example:

```
format => json_object('ignoremissingcolumns' value 'true', 'removequotes' value 'true',
                          'dateformat' value 'YYYY-MM-DD-HH24-MI-SS', 'blankasnull'
value 'true', 'logretention' value 7)
```

> **Note:**
>
> For Avro, ORC, or Parquet format options, see DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet.

As noted in the **Format Option** column, a limited set of format options are valid with `DBMS_CLOUD.COPY_COLLECTION` or with `DBMS_CLOUD.COPY_DATA` when the `format type` is JSON.

Format Option	Description	Syntax
`access_protocol`	Specifies the type of Apache Iceberg table, such as AWS or OCI Object Storage, and what information is used to create the external table, for example information from a data catalog or from a direct metadata URI.	See CREATE_EXTERNAL_TABLE Procedure for Apache Iceberg, for details on the `access_protocol` syntax.
`blankasnull`	When set to `true`, loads fields consisting of spaces as null.	`blankasnull : true` Default value: `False`
`characterset` Valid with `format` JSON and `COPY_DATA`	Specifies the characterset of source files	`characterset`: *string* Default value: Database characterset
`columnpath` Only use with `format` JSON and `COPY_DATA`	Array of JSON path expressions that correspond to the fields that need to be extracted from the JSON records. Each of the JSON path expressions in the array should follow the rules described in SQL/JSON Path Expressions . Only use with format JSON and `DBMS_CLOUD.COPY_DATA`.	JSON Array of json path expressions expressed in string format. For example: 'columnpath' value `'["$.WEATHER_STATION_ID", "$.WEATHER_STATION_NAME"]'`

Format Option	Description	Syntax
`compression` Option valid with JSON data	Specifies the compression type of the source file. ZIP archiving format is not supported. Specifying the value `auto` checks for the compression types: `gzip, zlib, bzip2`.	`compression: auto|gzip|zlib|bzip2` Default value: Null value meaning no compression.
`conversionerrors`	If a row is rejected because of data type conversion errors, the related columns are stored as null or the row is rejected.	`conversionerrors : reject_record | store_null` Default value: `reject_record`
`dateformat`	Specifies the date format in the source file. The format option `AUTO` searches for the following formats: `J` `MM-DD-YYYYBC` `MM-DD-YYYY` `YYYYMMDD HHMISS` `YYMMDD HHMISS` `YYYY.DDD` `YYYY-MM-DD`	`dateformat : string` Default value: Database date format
`delimiter`	Specifies the field delimiter. To use a special character as the delimiter, specify the HEX value of the ASCII code of the character. For example, the following specifies the TAB character as the delimiter: `format => json_object('delimiter' value 'X''9''')`	`delimiter : character` Default value `|` (pipe character)

Format Option	Description	Syntax
detectfieldorder	Specifies that the fields in the external data files are in a different order than the columns in the table. Detect the order of fields using the first row of each external data file and map it to the columns of the table. The field names in external data files are compared in case insensitive manner with the names of the table columns. This format option is applicable for the following procedures: • DBMS_CLOUD.COPY_DATA • DBMS_CLOUD.CREATE_EXTERNAL_TABLE • DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE • DBMS_CLOUD.CREATE_HYBRID_PART_TABLE Restrictions for detectfieldorder: • Field names in the data file must appear in the first record line and must not contain any white spaces between the field names. • The field delimiter in the field names record must be the same as the field delimiter for the data in the file. • Quoted field names are not supported. The field names in data files are compared, in case insensitive manner, with the names of the external table columns. • Embedded field delimiters are not allowed in the field names. • The number of columns in the table must match the number of fields in the data files. • This format option is not applicable for Bigdata or Oracle Data Pump formats, as those formats have precise column metadata information in the binary file format. The text formats, CSV, JSON, Parquet, or XML can benefit from this automatic field order detection when the first line contains the field names. See FIELD NAMES and the description for ALL FILES for more information.	detectfieldorder: true Default value: false
enablelogs	The format option enablelogs is used with the following DBMS_CLOUD procedures: • COPY_DATA • COPY_COLLECTION • EXPORT_DATA enablelogs specifies a boolean value, when set to TRUE, logs are generated. When set to FALSE, logs are not generated. For example: format => JSON_OBJECT('enablelogs' value FALSE)	enablelogs: false Default value: true

Format Option	Description	Syntax
`encryption`	The format option `encryption` specifies the encryption and decryption options to export and import data to and from the Object Store. Use `encryption` to specify the following parameters to encrypt and decrypt: • `user_defined_function`: Specifies a fully qualified user defined function to decrypt or encrypt the specified BLOB (binary large object). It returns a decrypted or encrypted BLOB. `user_defined_function` is mutually exclusive with other parameters for `encryption`. For example, `ADMIN.DECRYPTION_CALLBACK`. • `type`: Specifies the `DBMS_CRYPTO` encryption algorithm to decrypt or encrypt. `type` accepts values in the *Block Cipher Algorithms* + *Block Cipher Chaining Modifiers* + *Block Cipher Padding Modifiers* format. Supported Block Cipher Algorithms are: – `DBMS_CRYPTO.ENCRYPT_AES256` Supported Block Cipher Chaining Modifiers are: – `DBMS_CRYPTO.CHAIN_CBC` – `DBMS_CRYPTO.CHAIN_CFB` – `DBMS_CRYPTO.CHAIN_ECB` – `DBMS_CRYPTO.CHAIN_OFB` Supported Block Cipher Padding Modifiers are: – `DBMS_CRYPTO.PAD_PKCS5` – `DBMS_CRYPTO.PAD_NONE` – `DBMS_CRYPTO.PAD_ZERO` – `DBMS_CRYPTO.PAD_ORCL` • `credential_name`: Specifies the credential used to store the encryption key. The Block Cipher Chaining Modifiers and Block Cipher Padding Modifiers values defaults to `DBMS_CRYPTO.CHAIN_CBC` and `DBMS_CRYPTO.PAD_PKCS5`, if you do not specify values for these parameters. The format option `encryption` is used with the following `DBMS_CLOUD` procedures: • Used to pass parameters to decrypt for these procedures: – `DBMS_CLOUD.COPY_DATA` – `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` – `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` – `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` For `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` this option is only applicable to the Object Storage files. – `DBMS_CLOUD.COPY_COLLECTION`	`encryption:`*value* Where *value* is a JSON string that provides additional parameters for encryption: `type`: *value* Specifies the encryption type. `credential_name`: *value* Specifies the credential used to store the encryption key. `user_defined_function`: *value* Specifies a fully qualified user-defined function to decrypt or encrypt the specified BLOB (binary large object).

Format Option	Description	Syntax
	• Used to pass parameters to encrypt for these procedure: – DBMS_CLOUD.EXPORT_DATA For example: format => JSON_OBJECT('encryption' value json_object ('type' value DBMS_CRYPTO.ENCRYPT_AES256 + DBMS_CRYPTO.CHAIN_CBC + DBMS_CRYPTO.PAD_PKCS5, 'credential_name' value 'ENCRYPTION_CRED'))	
endquote	Data can be enclosed between two delimiters, specified with quote and endquote. The quote and endquote characters are removed during loading when specified. For example: format => JSON_OBJECT('quote' value '(', 'endquote' value ')')	endquote:character Default value: Null, meaning no endquote.
escape	The character "\" is used as the escape character when specified.	escape : true Default value: false
ignoreblanklines Option valid with JSON data	Blank lines are ignored when set to true.	ignoreblanklines : true Default value: False
ignoremissingcolumns	If there are more columns in the field_list than there are in the source files, the extra columns are stored as null.	ignoremissingcolumns : true Default value False
jsonpath Only use with COPY_COLLECTION	JSON path to identify the document to load. This option is valid only for JSON collection data with DBMS_CLOUD.COPY_COLLECTION.	jsonpath: *string* Default value: Null
keyassignment Only use with COPY_COLLECTION	Specifies whether a new collection is created as a mongo-compatible collection or as a SODA collection. When the value is set to embedded_oid, a new collection is created as a mongo-compatible collection. By default this parameter is not set, meaning a new collection is created as a SODA collection.	keyassignment: embedded_oid Default: keyassignment is not set
keypath Only use with COPY_COLLECTION	Specifies an attribute in the data to be loaded as the '_id' value. If keypath is specified then you must also specify the keyassignment value as embedded_oid. Set the value to a path, for example, '$.mykey', to pick the value of the path as '_id' value. This parameter is optional and is only valid for loading into mongo-compatible collections. If not specified, Oracle generates a 12-byte unique system ID and populates that as the '_id' attribute, if an '_id' attribute is not already present in the data being loaded.	keypath: *string* Default: keypath is not set. When keypath is set, the default *string* value is NULL.

Format Option	Description	Syntax
`language`	Specifies a language name (for example, FRENCH), from which locale-sensitive information can be derived.	`language`: *string* Default value: Null See Locale Data in *Oracle Database Globalization Support Guide* for a listing of Oracle-supported languages.
`logdir`	Specifies a string value that determines the directory object name where the `logfile_table` or `badfile_table` files are saved. By default, the `logdir` is not case-sensitive, but the case is reserved when the specified value is enclosed in double-quotes. For example: `format => JSON_OBJECT ('logdir' value 'test_log')` The `logdir` format option specified in the above example saves the `logfile_table` or `badfile_table` files in the `TEST_LOG` directory object. `format => JSON_OBJECT ('logdir' value '"test_log"')` The `logdir` format option specified in the above example saves the `logfile_table` or `badfile_table` files in the `test_log` directory object.	`logdir`: *string* Default value: `DATA_PUMP_DIR`
`logprefix`	Specifies a string value that determines the prefix for the `logfile_table` and `badfile_table` files. The log table name format is: *logprefix*`$`*operation_id* By default, the logprefix is in upper case, but the case is reserved when the specified value is enclosed in double-quotes. For example: `format => JSON_OBJECT ('logprefix' value 'TEST')` Log files then use the `TEST` prefix, such as: `TEST$2_LOG` and `TEST$2_BAD`.	`logprefix`: *string* Default value: `COPY`
`logretention`	Specifies a positive integer duration, in days, for which the `logfile_table` and `badfile_table` files are retained. Valid values: `0` to `99999` For example: `format => JSON_OBJECT ('logretention' value 7)`	`logretention`: *number* Default value: `2`
`maxdocsize` This option is valid only with JSON data	Maximum size of JSON documents.	`maxdocsize`: *number* Default value: 1 Megabyte Maximum allowed value: 2 Gigabytes

Format Option	Description	Syntax
`numericcharacters`	Specifies the characters to use as the group separator and decimal character. *decimal_character*: The decimal separates the integer portion of a number from the decimal portion. *group_separator*: The group separator separates integer groups (that is, thousands, millions, billions, and so on).	`numericcharacters: '`*`decimal_character group_separator`*`'` Default value: `".,"` See NLS_NUMERIC_CHARACTERS in *Oracle Database Globalization Support Guide* for more information.
`numberformat`	Specifies the number format model. Number format models cause the number to be rounded to the specified number of significant digits. A number format model is composed of one or more number format elements. This is used in combination with `numericcharacters`.	`numberformat:` *`number_format_model`* Default value: is derived from the setting of the `NLS_TERRITORY` parameter See Number Format Models in *SQL Language Reference* for more information.

Format Option	Description	Syntax
partition_columns	The format option partition_columns is used with DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE to specify the column names and data types of partition columns when the partition columns are derived from the file path, depending on the type of data file, structured or unstructured: • When the DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE includes the column_list parameter and the data file is unstructured, such as with CSV text files, partition_columns does not include the data type. For example, use a format such as the following for this type of partition_columns specification: '"partition_columns": ["state","zipcode"]' The data type is not required because it is specified in the DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE column_list parameter. • When the DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE does not include the column_list parameter and the data files are structured, such as Avro, ORC, or Parquet files, the partition_columns option includes the data type. For example, the following shows a partition_columns specification: '"partition_columns":[{"name":"country", "type":"varchar2(10)"}, {"name":"year", "type":"number"}, {"name":"month", "type":"varchar2(10)"}]' If the data files are unstructured and the type sub-clause is specified with partition_columns, the type sub-clause is ignored. For object names that are not based on hive format, the order of the partition_columns specified columns must match the order as they appear in the object name in the file_uri_list.	
quote	Specifies the quote character for the fields, the quote characters are removed during loading when specified.	quote: *character* Default value: Null meaning no quote

Format Option	Description	Syntax
`recorddelimiter` Option valid with JSON data	Specifies the record delimiter. By default, `DBMS_CLOUD` tries to automatically find the correct newline character as the delimiter. It first searches the file for the Windows newline character "`\r\n`". If it finds the Windows newline character, this is used as the record delimiter for all files in the procedure. If a Windows newline character is not found, it searches for the UNIX/Linux newline character "`\n`" and if it finds one it uses "`\n`" as the record delimiter for all files in the procedure. Specify this argument explicitly if you want to override the default behavior, for example: `format => json_object('recorddelimiter' VALUE '''\r\n''')` To indicate that there is no record delimiter you can specify a `recorddelimiter` that does not occur in the input file. For example, to indicate that there is no delimiter, specify the control character 0x01 (SOH) as a value for the `recorddelimiter` and set the `recorddelimiter` value to "`0x''01''`" (this character does not occur in JSON text). For example: `format => '{"recorddelimiter" : "0x''01''"}'` The `recorddelimiter` is set once per procedure call. If you are using the default value, `detected newline`, then all files use the same record delimiter, if one is detected.	`recorddelimiter`: *character* Default value: detected newline

Format Option	Description	Syntax
`regexuri`	The format option `regexuri` is used with the following DBMS_CLOUD procedures: • `COPY_COLLECTION` • `COPY_DATA` • `CREATE_EXTERNAL_TABLE` • `CREATE_EXTERNAL_PART_TABLE` • `CREATE_HYBRID_PART_TABLE` When the value of `regexuri` is set to `TRUE`, you can use wildcards as well as regular expressions in the file names in Cloud source file URIs. The characters "*" and "?" are considered wildcard characters when the `regexuri` parameter is set to `FALSE`. When the `regexuri` parameter is set to `TRUE` the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the `REGEXP_LIKE` function. Regular expression patterns are not supported for directory names. For external tables, this option is only supported with the tables that are created on a file in the Object Storage. For example: `format => JSON_OBJECT('regexuri' value TRUE)` See REGEXP_LIKE Condition for more information on `REGEXP_LIKE` condition.	`regexuri`: `True` Default value : `False`
`rejectlimit`	The operation will error out after specified number of rows are rejected.	`rejectlimit`: *number* Default value: `0`
`removequotes`	Removes any quotes that are around any field in the source file.	`removequotes`: `true` Default value: `False`
`skipheaders`	Specifies how many rows should be skipped from the start of the file.	`skipheaders`: *number* Default value: 0 if not specified, 1 if specified without a value
`territory`	Specifies a territory name to further determine input data characteristics.	`territory`: *string* Default value: Null See Locale Data in *Oracle Database Globalization Support Guide* for a listing of Oracle-supported territories.
`timestampformat`	Specifies the timestamp format in the source file. The format option `AUTO` searches for the following formats: `YYYY-MM-DD HH:MI:SS.FF` `YYYY-MM-DD HH:MI:SS.FF3` `YYYY-MM-DD HH24:MI:SS.FF3` `MM/DD/YYYY HH:MI:SS.FF3`	`timestampformat` : *string* Default value: Database timestamp format The string can contain wildcard characters such as "$".

Format Option	Description	Syntax
`timestampltzformat`	Specifies the timestamp with local timezone format in the source file. The format option `AUTO` searches for the following formats: `DD Mon YYYY HH:MI:SS.FF TZR` `MM/DD/YYYY HH:MI:SS.FF TZR` `YYYY-MM-DD HH:MI:SS+/-TZR` `YYYY-MM-DD HH:MI:SS.FF3` `DD.MM.YYYY HH:MI:SS TZR`	`timestampltzformat` : *string* Default value: Database timestamp with local timezone format
`timestamptzformat`	Specifies the timestamp with timezone format in the source file. The format option `AUTO` searches for the following formats: `DD Mon YYYY HH:MI:SS.FF TZR` `MM/DD/YYYY HH:MI:SS.FF TZR` `YYYY-MM-DD HH:MI:SS+/-TZR` `YYYY-MM-DD HH:MI:SS.FF3` `DD.MM.YYYY HH:MI:SS TZR`	`timestamptzformat`: *string* Default value: Database timestamp with timezone format
`trimspaces`	Specifies how the leading and trailing spaces of the fields are trimmed. See the description of trim_spec.	`trimspaces`: `rtrim| ltrim| notrim| lrtrim| ldrtrim` Default value: `notrim`
`truncatecol`	If the data in the file is too long for a field, then this option will truncate the value of the field rather than reject the row.	`truncatecol`:`true` Default value: `False`

Format Option	Description	Syntax
`type`	Specifies the source file type. See the description of `CSV` in field_definitions Clause If the `type` is `datapump`, then the only other valid format option is `rejectlimit`. If the `type` is `datapump`, then the only Object Stores supported are Oracle Cloud Infrastructure Object Storage and Oracle Cloud Infrastructure Object Storage Classic. See DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet for `type` values `avro`, `orc`, or `parquet`. For JSON data with `DBMS_CLOUD.COPY_COLLECTION` `type` has two valid values: `json` (default) and `ejson`. For `DBMS_CLOUD.COPY_COLLECTION` these values both specify that the input is JSON data. The value `ejson` causes extended objects in the textual JSON input data to be translated to scalar JSON values in the native binary JSON collection. The value `json` does not perform this transformation and all objects in the input data are converted to binary JSON format. For JSON data with `DBMS_CLOUD.COPY_DATA` `type` has one valid value: `json`. This value specifies that the input is JSON data. • See COPY_COLLECTION Procedure for information on loading JSON objects. • See Textual JSON Objects That Represent Extended Scalar Values for information about extended JSON objects, which are interpreted when parameter `type` has value `ejson`.	`type: csv|csv with embedded|csv without embedded |avro||datapump| orc|parquet` **Note:** Not all `DBMS_CLOUD` procedures support all of these types. `csv` is the same as `csv without embedded`. Default value: Null For JSON data there are two valid `type` values for use with `DBMS_CLOUD.COPY_COLLECTION`: `json|ejson` In this case the default value is `json`. For JSON data with `DBMS_CLOUD.COPY_DATA`, only `json` is valid.
`unpackarrays` Only use with `COPY_COLLECTION`	When set to `true`, if a loaded document is an array, then the contents of the array are loaded as documents rather than the array itself. This only applies to the top-level array. When set to `true`, the entire array is inserted as a single document. This option is valid only for JSON collection data with `DBMS_CLOUD.COPY_COLLECTION`.	`unpackarrays: true` Default value: `False`

DBMS_CLOUD Package Format Options for EXPORT_DATA

Describes the valid format parameter options for `DBMS_CLOUD.EXPORT_DATA` with text file formats, CSV, JSON, Parquet, or XML, and for Oracle Data Pump.
These are the valid `format` parameters for use with `DBMS_CLOUD.EXPORT_DATA`. You specify text file output when you use the `format type` option and the value is one of: `csv`, `json`, `parquet`, or `xml`. This also shows the `format` options when the `format type` is `datapump`.

The two ways to specify the format argument are:

```
format => '{"format_option" : "format_value" }'
```

And:

```
format => json_object('format_option' value 'format_value'))
```

Examples:

```
format => json_object('type' VALUE 'json')
```

To specify multiple format options, separate the values with a ",".

For example:

```
format => json_object('compression' value 'gzip', 'type' value 'json')
```

This table covers the format options for `DBMS_CLOUD.EXPORT_DATA` when the `format` parameter `type` option is one of: CSV, JSON, Parquet, or XML. For other procedures and other output types, see DBMS_CLOUD Package Format Options for the list of format options.

Format Option	Description	Syntax
`compression`	Specifies the compression type of the source file. Note: ZIP archiving format is not supported. When the `format type` is `csv`, `json`, or `xml`, the default compression is Null, meaning no compression. When the `format type` is `parquet`, the default compression is `snappy`. When the `format type` is `datapump` you can specify supported Oracle Data Pump access parameters: • `compression`: The valid values are: `BASIC`, `LOW`, `MEDIUM`, and `HIGH`. • `version`: The valid values are: `COMPATIBLE`, `LATEST`, and a specified *`version_number`*.	When the `type` is:`csv` | `json` | `xml` `compression:gzip` Default value: Null value meaning no compression. When the `type` is `parquet` `compression: gzip` | `snappy` Default value: `snappy` When the type is `datapump` `compression: BASIC|LOW|MEDIUM|HIGH`
`delimiter`	Specifies a custom field delimiter. `format => json_object('delimiter' value '|')` The delimiter value cannot be an ASCII code or an escape character. **Note:** This option only applies with `csv type`.	`delimiter` : *character* Default value , (comma)
`endquote`	Specifies that fields can be enclosed between two delimiters, with `quote` and `endquote`. If `endquote` is not specified, then the `quote` character will be used by default as the `endquote` character. For example: `format => JSON_OBJECT('quote' value '(', 'endquote' value ')')` **Note:** This option only applies with `csv type`.	`endquote`:*character* Default value: Null, meaning no `endquote`.

Format Option	Description	Syntax
`escape`	Specifies the occurrence of quote character in the field value using `"\"` character.	`escape : true` Default value: `false`

> **Note:**
>
> This option only applies with `csv type`.

Format Option	Description	Syntax
`encryption`	The format option `encryption` specifies the encryption and decryption options to export and import data to and from the Object Store. Use `encryption` to specify the following parameters to encrypt and decrypt: • `user_defined_function`: Specifies a fully qualified user-defined function to decrypt or encrypt the specified BLOB (binary large object). It returns a decrypted or encrypted BLOB. This parameter is mutually exclusive with other parameters for `encryption`. For example, `ADMIN.DECRYPTION_CALLBACK`. • `type`: Specifies the built-in encryption algorithm to decrypt or encrypt. `user_defined_function` and `type` are mutually exclusive. `type` accepts values in the *Block Cipher Algorithms* + *Block Cipher Chaining Modifiers* + *Block Cipher Padding Modifiers* format. Supported Block Cipher Algorithms are: – `DBMS_CRYPTO.ENCRYPT_AES256` Supported Block Cipher Chaining Modifiers are: – `DBMS_CRYPTO.CHAIN_CBC` – `DBMS_CRYPTO.CHAIN_CFB` – `DBMS_CRYPTO.CHAIN_ECB` – `DBMS_CRYPTO.CHAIN_OFB` Supported Block Cipher Padding Modifiers are: – `DBMS_CRYPTO.PAD_PKCS5` – `DBMS_CRYPTO.PAD_NONE` – `DBMS_CRYPTO.PAD_ZERO` – `DBMS_CRYPTO.PAD_ORCL` • `credential_name`: Specifies the credential used to store the encryption key. The Block Cipher Chaining Modifiers and Block Cipher Padding Modifiers values defaults to `DBMS_CRYPTO.CHAIN_CBC` and `DBMS_CRYPTO.PAD_PKCS5`, if you do not specify values for these parameters. The format option `encryption` is used with the following `DBMS_CLOUD` procedures: • Used to pass parameters to decrypt for these procedures: – `DBMS_CLOUD.COPY_DATA` – `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` – `DBMS_CLOUD.CREATE_HYBRID_TABLE` – `DBMS_CLOUD.COPY_COLLECTION` • Used to pass parameters to encrypt for these procedure: – `DBMS_CLOUD.EXPORT_DATA` For example: `format => JSON_OBJECT('encryption' value json_object ('type' value DBMS_CRYPTO.ENCRYPT_AES256 +`	`encryption:`*value* Where *value* is a JSON string that provides additional parameters for encryption: `type`: *value* Specifies the encryption type. `credential_name`: *value* Specifies the credential used to store the encryption key. `user_defined_function`: *value* Specifies a fully qualified user-defined function to decrypt or encrypt the specified BLOB (binary large object).

Format Option	Description	Syntax
	`DBMS_CRYPTO.CHAIN_CBC + DBMS_CRYPTO.PAD_PKCS5, 'credential_name' value 'ENCRYPTION_CRED'))`	
`header`	Writes column names as the first line in output files of `csv` type. The `header` option can accept a `boolean` or a `string` value. The valid values are: • `false`: Skips the header row. • `true`: Includes the header row. The column names are based on the `SELECT` statement in the `query` parameter. You must specify column aliases in the `SELECT` statement when using virtual columns or expressions. • *`String to define custom header names`*: Enables you to define header rows with custom names. The number of columns and delimiters in the string value must match the number of columns and delimiters in the `SELECT` statement. The default delimiter is comma (,). For example: `format => JSON_OBJECT('type' value 'csv', 'delimiter' value '|', 'compression' value 'gzip', 'header' value true)` **Note:** This option only applies with `csv type`.	`header: true| false| String to define custom header names` Default value: `false`
`fileextension`	Custom file extension to override the default choice for the format type. This applies to text formats with `DBMS_CLOUD.EXPORT_DATA`: CSV, JSON, Parquet, or XML. If the specified string does not start with period (dot), then a dot is automatically inserted before the file extension in the final file name. If no file extension is desired, use the value: `fileextension ='none'`	Valid values: Any file extension. Default value: Depends on the `format type` option: • CSV format: `.csv` • JSON format: `.json` • PARQUET format: `.parquet` • XML format: `.xml`
`maxfilesize`	Number in bytes for maximum size of output generated. This applies to text based formats for exporting data with `DBMS_CLOUD.EXPORT_DATA` when the `format type` option is set to,`csv`, `json`, or `xml`. Note: This option is not valid when the `format type` option is `parquet`.	Minimum value: `10485760` (10 MB) Maximum value: 1 GB Default value: `10485760` (10 MB)

Format Option	Description	Syntax
`quote`	In CSV format, fields can be enclosed between two delimiters. Specify the delimiters with `quote` and `endquote`. If `endquote` is not specified, then the `quote` character will be used by default as the `endquote` character. **Note:** This option only applies with `csv type`.	`quote`: *character* Default value: Null meaning do not enclose fields with quotes.
`trimspaces`	Specifies how the leading and trailing spaces of the fields are trimmed for CSV format. Trim spaces is applied before quoting the field, if the `quote` parameter is specified. See the description of trim_spec. **Note:** This option only applies with `csv type`.	`trimspaces: rtrim| ltrim| notrim| lrtrim| ldrtrim` Default value: `notrim`
`type`	Specifies the output file type. `csv`: specifies Character Separated Values (CSV) format that allows you to export query results as a set of column values separated by any custom character. `json`: specifies to export the query results as JSON files. `parquet`: specifies to export the query results as Parquet files. `xml`: specifies to export query results as rows of valid XML documents. Each row is encapsulated in a root XML tag of `<RECORD> </RECORD>`. The query result is automatically transformed into XML format using XMLFOREST SQL function. Use Column Aliases to customize the XML tag names for columns. When the `format type` is `datapump` you can specify supported Oracle Data Pump access parameters: • `compression`: The valid values are: `BASIC`, `LOW`, `MEDIUM`, and `HIGH`. • `version`: The valid values are: `COMPATIBLE`, `LATEST`, and a specified *`version_number`*. See access_parameters Clause for more information.	`type: csv|datapump|json| parquet|xml`

DBMS_CLOUD Avro, ORC, and Parquet Support

This section covers the `DBMS_CLOUD` Avro, ORC, and Parquet support provided with Autonomous Database.

DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet

The format argument in DBMS_CLOUD specifies the format of source files.

The two ways to specify the format argument are:

```
format => '{"format_option" : "format_value" }'
```

And:

```
format => json_object('format_option' value 'format_value'))
```

Examples:

```
format => json_object('type' VALUE 'CSV')
```

To specify multiple format options, separate the values with a ",".

For example:

```
format => json_object('ignoremissingcolumns' value 'true', 'removequotes' value 'true',
'dateformat' value 'YYYY-MM-DD-HH24-MI-SS', 'blankasnull' value 'true')
```

Format Option	Description	Syntax
regexuri	When the value of regexuri is set to TRUE, you can use wildcards as well as regular expressions in the file names in Cloud source file URIs. The characters "*" and "?" are considered wildcard characters when the regexuri parameter is set to FALSE. When the regexuri parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the REGEXP_LIKE function. Regular expression patterns are not supported for directory names. For external tables, this option is only supported with the tables that are created on a file in the Object Storage. For example: format => JSON_OBJECT('regexuri' value TRUE) See REGEXP_LIKE Condition for more information on REGEXP_LIKE condition.	regexuri: True Default value : False
type	Specifies the file type.	type : avro | orc | parquet
schema	When schema is set to first or all, the external table columns and data types are automatically derived from the Avro, ORC, or Parquet file metadata. The column names will match those found in Avro, ORC, or Parquet. The data types are converted from Avro, ORC, or Parquet data types to Oracle data types. All columns are added to the table. The value first specifies to use the metadata from the first Avro, ORC, or Parquet file in the file_uri_list to auto generate the columns and their data types. Use first if all of the files have the same schema. The value all specifies to use the metadata from all Avro, ORC, or Parquet files in the file_uri_list to auto generate the columns and their data types. Use all (slower) if the files may have different schemas. **Default**: If column_list is specified, then the schema value, if specified is ignored. If column_list is not specified then the schema default value is first. **Note**: For Avro, ORC, or Parquet format files the schema format option is not available and the column_list parameter must be specified for partitioned external tables using the DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE procedure.	schema : first | all

DBMS_CLOUD Package Avro to Oracle Data Type Mapping

Describes the mapping of Avro data types to Oracle data types.

> **Note:**
>
> Complex types, such as maps, arrays, and structs are supported starting with Oracle Database 19c. See DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types for information on using Avro complex types.

Avro Type	Oracle Type
INT	NUMBER(10)
LONG	NUMBER(19)
BOOL	NUMBER(1)
UTF8 BYTE_ARRAY	RAW(2000)
FLT	BINARY_FLOAT
DBL	BINARY_DOUBLE
DECIMAL(p)	NUMBER(p)
DECIMAL(p,s)	NUMBER(p,s)
DATE	DATE
STRING	VARCHAR2
TIME_MILLIS	VARCHAR2(20 BYTE)
TIME_MICROS	VARCHAR2(20 BYTE)
TIMESTAMP_MILLIS	TIMESTAMP(3)
TIMESTAMP_MICROS	TIMESTAMP(6)
ENUM	VARCHAR2(*n*) Where: "*n*" is the actual maximum length of the AVRO ENUM's possible values
DURATION	RAW(2000)
FIXED	RAW(2000)
NULL	VARCHAR2(1) BYTE

See DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types for information on using Avro complex types.

DBMS_CLOUD Package ORC to Oracle Data Type Mapping

Describes the mapping of ORC data types to Oracle data types.

See DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types for information on using ORC complex types.

ORC Type	Oracle Type	More Information
array	VARCHAR2(*n*) JSON format	DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types
bigint (64 bit)	NUMBER(19)	
binary	BLOB	
boolean (1 bit)	NUMBER(1)	

ORC Type	Oracle Type	More Information
char	CHAR(*n*)	
date	DATE	
double	BINARY_DOUBLE	
float	BINARY_FLOAT	
int (32 bit)	NUMBER(10)	
list	VARCHAR2(*n*) JSON format	DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types
map	VARCHAR2(*n*) JSON format	DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types
smallint (16 bit)	NUMBER(5)	
string	VARCHAR2(4000)	
struct	VARCHAR2(*n*) JSON format	DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types
timestamp	TIMESTAMP	
tinyint (8 bit)	NUMBER(3)	
union	VARCHAR2(*n*) JSON format	DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types
varchar	VARCHAR2(*n*)	

DBMS_CLOUD Package Parquet to Oracle Data Type Mapping

Describes the mapping of Parquet data types to Oracle data types.

> **Note:**
>
> Complex types, such as maps, arrays, and structs are supported starting with Oracle Database 19c. See DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types for information on using Parquet complex types.

Parquet Type	Oracle Type
UINT_64	NUMBER(20)
INT_64	NUMBER(19)
UINT_32	NUMBER(10)
INT_32	NUMBER(10)
UINT_16	NUMBER(5)
INT_16	NUMBER(5)
UINT_8	NUMBER(3)
INT_8	NUMBER(3)
BOOL	NUMBER(1)
UTF8 BYTE_ARRAY	VARCHAR2(4000 BYTE)
FLT	BINARY_FLOAT
DBL	BINARY_DOUBLE
DECIMAL(p)	NUMBER(p)

Parquet Type	Oracle Type
DECIMAL(p,s)	NUMBER(p,s)
DATE	DATE
STRING	VARCHAR2(4000)
TIME_MILLIS	VARCHAR2(20 BYTE)
TIME_MILLIS_UTC	VARCHAR2(20 BYTE)
TIME_MICROS	VARCHAR2(20 BYTE)
TIME_MICROS_UTC	VARCHAR2(20 BYTE)
TIMESTAMP_MILLIS	TIMESTAMP(3)
TIMESTAMP_MILLIS_UTC	TIMESTAMP(3)
TIMESTAMP_MICROS	TIMESTAMP(6)
TIMESTAMP_MICROS_UTC	TIMESTAMP(6)
TIMESTAMP_NANOS	TIMESTAMP(9)

See DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types for information on using Parquet complex types.

DBMS_CLOUD Package Oracle Data Type to Parquet Mapping

Describes the mapping of Oracle data types to Parquet data types.

Oracle Type	Parquet Type
BINARY_DOUBLE	DBL
BINARY_FLOAT	FLT
DATE	DATE
NUMBER(p,s)	DECIMAL(p,s)
NUMBER(p)	DECIMAL(p)
TIMESTAMP(3)	TIMESTAMP_MILLIS
TIMESTAMP(3)	TIMESTAMP_MILLIS_UTC
TIMESTAMP(6)	TIMESTAMP_MICROS
TIMESTAMP(6)	TIMESTAMP_MICROS_UTC
TIMESTAMP(9)	TIMESTAMP_NANOS
VARCHAR2(4000)	STRING

NLS Session Parameters

The NLS session parameters `NLS_DATE_FORMAT`, `NLS_TIMESTAMP_FORMAT`, `NLS_TIMESTAMP_TZ_FORMAT` and `NLS_NUMERIC_CHARACTERS` define how the date, timestamp, timestamp with time zone format, and radix separator for timestamp with decimal marker should be shown when a table with those column types are queried.

In addition, when you export data using `DBMS_CLOUD.EXPORT_DATA` and specify Parquet output, Autonomous Database reads the values of these parameters from the `NLS_SESSION_PARAMETERS` table. Autonomous Database uses these values to convert the Oracle data types `DATE` or `TIMESTAMP` to Parquet types.

The `NLS_SESSION_PARAMETERS` parameters support an `RR` format mask (two character year specification).

The RR format mask for the year is not supported for these parameters when you export data to Parquet with DBMS_CLOUD.EXPORT_DATA. An application error is raised if you attempt to export to parquet and the NLS_SESSION_PARAMETERS are set to use the RR format mask (the default value for the RR format depends on the value of the NLS_TERRITORY parameter).

When one of the parameters NLS_DATE_FORMAT, NLS_TIMESTAMP_FORMAT or NLS_TIMESTAMP_TZ_FORMAT uses the RR format mask, you must change the format value to supported value to export data to Parquet with DBMS_CLOUD.EXPORT_DATA. For example:

```
ALTER SESSION SET NLS_DATE_FORMAT = "MM/DD/YYYY";
ALTER SESSION SET NLS_TIMESTAMP_FORMAT = 'YYYY-MM-DD HH:MI:SS.FF';
ALTER SESSION SET  NLS_TIMESTAMP_TZ_FORMAT='YYYY-MM-DD HH:MI:SS.FF TZH:TZM';
```

After you change the value, you can verify the change by querying the NLS_SESSION_PARAMETERS view:

```
SELECT value FROM NLS_SESSION_PARAMETERS
       WHERE parameter IN
('NLS_DATE_FORMAT','NLS_TIMESTAMP_FORMAT','NLS_TIMESTAMP_TZ_FORMAT');
```

If NLS_DATE_FORMAT is set, it applies to the columns with DATE datatype. If NLS_TIMESTAMP_FORMAT is set, it applies to the columns with TIMESTAMP datattype. If NLS_TIMESTAMP_TZ_FORMAT is set, it applies to the columns with TIMESTAMP WITH TIME ZONE datatype.

See Date and Time Parameters and NLS Data Dictionary Views for more information.

DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types

Describes the mapping of Avro, ORC, and Parquet complex data types to Oracle data types.

Autonomous Database supports complex data types, including the following complex types:

- struct
- list
- map
- union
- array

When you specify a source file type of Avro, ORC, or Parquet and the source file includes complex columns, Autonomous Database queries return JSON for the complex columns. This simplifies processing of query results; you can use Oracle's powerful JSON parsing features consistently across the file types and data types. The following table shows the format for the complex types in Autonomous Database:

> **Note:**
>
> The complex fields map to VARCHAR2 columns and VARCHAR2 size limits apply.

Type	Parquet	ORC	Avro	Oracle
List: sequence of values	List	List	Array	VARCHAR2 (JSON format)
Map: list of objects with single key	Map	Map	Map	VARCHAR2 (JSON format)
Union: values of different type	Not Available	Union	Union	VARCHAR2 (JSON format)
Object: zero or more key-value pairs	Struct	Struct	Record	VARCHAR2 (JSON format)

If your ORC, Parquet, or Avro source files contain complex types, then you can query the JSON output for these common complex types. For example, the following shows an ORC file, `movie-info.orc`, with a complex type (the same complex type handling applies for Parquet and Avro source files).

Consider the `movie-info.orc` file with the following schema:

```
id    int
original_title string
overview       string
poster_path    string
release_date   string
vote_count     int
runtime        int
popularity     double
genres         array<struct<id:int,name:string>
```

Notice that each movie is categorized by multiple `genres` using an array of `genres`. The `genres` array is an `array` of `structs` and each item has an `id` (`int`) and a `name` (`string`). The `genres` array is considered a complex type. You can create a table over this ORC file using `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` as follows:

```
BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
        table_name =>'movie_info',
        credential_name =>'OBJ_STORE_CRED',
        file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
mytenancy/b/movies/o/movie-info.orc',
        format => '{"type":"orc", "schema": "first"}');
END;
/
```

When you create the external table the database automatically generates the columns based on the schema in the ORC file (if you are using Avro or Parquet, the same applies). For this example, the `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` creates a table in your database as follows:

```
CREATE TABLE "ADMIN"."MOVIE_INFO"
    ( "ID"
      NUMBER(10,0),
      "ORIGINAL_TITLE"  VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
      "OVERVIEW"        VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
```

```
      "POSTER_PATH"     VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
      "RELEASE_DATE"    VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
      "VOTE_COUNT"      NUMBER(10,0),
      "RUNTIME"         NUMBER(10,0),
      "POPULARITY"      BINARY_DOUBLE,
      "GENRES"          VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP"
    )  DEFAULT COLLATION "USING_NLS_COMP"
    ORGANIZATION EXTERNAL
     ( TYPE      ORACLE_BIGDATA
       DEFAULT DIRECTORY "DATA_PUMP_DIR"
       ACCESS PARAMETERS
       ( com.oracle.bigdata.credential.name=OBJ_STORE_CRED
         com.oracle.bigdata.fileformat=ORC
   )
       LOCATION
        (
      'https://objectstorage.us-phoenix-1.oraclecloud.com/n/mytenancy/b/
movies/o/movie-info.orc'
           )
     )
   REJECT LIMIT UNLIMITED
   PARALLEL;
 )
```

Now you can query the movie data:

```
SELECT original_title, release_date, genres
     FROM movie_info
     WHERE release_date > '2000'
     ORDER BY original_title;
```

This produces the following output:

```
original_title              release_date   genres
(500) Days of Summer        2009           [{"id":3,"name":"Drama"},
{"id":6,"name":"Comedy"},{"id":17,"name":"Horror"},{"id":19,"name":"Western"},
{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
10,000 BC                   2008           [{"id":6,"name":"Comedy"}]
11:14                       2003           [{"id":9,"name":"Thriller"},
{"id":14,"name":"Family"}]
127 Hours                   2010           [{"id":6,"name":"Comedy"},
{"id":3,"name":"Drama"}]
13 Going on 30              2004           [{"id":6,"name":"Comedy"},
{"id":3,"name":"Drama"},{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
1408                        2007           [{"id":45,"name":"Sci-Fi"},
{"id":6,"name":"Comedy"},{"id":17,"name":"Horror"},{"id":6,"name":"Comedy"},
{"id":18,"name":"War"}]
```

Notice that the complex type `genres` is returned as a JSON array.

To make the JSON data more useful, you can transform the column using Oracle's JSON functions. For example, you can use the JSON "." notation as well as the more powerful transform functions such as `JSON_TABLE`.

See Simple Dot-Notation Access to JSON Data for information on "." notation.

See SQL/JSON Function JSON_TABLE for information on `JSON_TABLE`.

The following example shows a query on the table that takes each value of the array and turns the value into a row in the result set:

```
SELECT original_title, release_date, m.genre_name, genres
    FROM movie_info mi,
       JSON_TABLE(mi.genres, '$.name[*]'
        COLUMNS (genre_name VARCHAR2(25) PATH
      '$')
                 ) AS m
 WHERE rownum < 10;
```

The `JSON_TABLE` creates a row for each value of the array, think outer join, and the struct is parsed to extract the name of the genre. This produces the following output:

```
original_title                  release_date        genre_name        genres
(500) Days of Summer            2009                Drama
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"},{"id":19,"name":"Western"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer            2009                Comedy
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"},{"id":19,"name":"Western"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer            2009                Horror
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"},{"id":19,"name":"Western"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer            2009                Western
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"},{"id":19,"name":"Western"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer            2009                War
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"},{"id":19,"name":"Western"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer            2009                Romance
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"},{"id":19,"name":"Western"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
```

```
10,000 BC                       2008                Comedy
[{"id":6,"name":"Comedy"}]
11:14                           2003                Family
[{"id":9,"name":"Thriller"},{"id":14,"name":"Family"}]
11:14                           2003                Thriller
[{"id":9,"name":"Thriller"},{"id":14,"name":"Family"}]
127 Hours                       2010                Comedy
[{"id":6,"name":"Comedy"},{"id":3,"name":"Drama"}]
127 Hours                       2010                Drama
[{"id":6,"name":"Comedy"},{"id":3,"name":"Drama"}]
13 Going on 30                  2004                Romance
[{"id":6,"name":"Comedy"},{"id":3,"name":"Drama"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
13 Going on 30                  2004                Comedy
[{"id":6,"name":"Comedy"},{"id":3,"name":"Drama"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
13 Going on 30                  2004                War
[{"id":6,"name":"Comedy"},{"id":3,"name":"Drama"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
13 Going on 30                  2004                Drama
[{"id":6,"name":"Comedy"},{"id":3,"name":"Drama"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
```

DBMS_CLOUD Package Avro, ORC, and Parquet to Oracle Column Name Mapping

Describes rules for how Avro, ORC, and Parquet column names are converted to Oracle column names.

The following are supported for Avro, ORC, and Parquet column names, but may require use of double quotes for Oracle SQL references in external tables. Thus, for ease of use and to avoid having to use double quotes when referencing column names, if possible do not use the following in Avro, ORC, and Parquet column names:

- Embedded blanks
- Leading numbers
- Leading underscores
- Oracle SQL reserved words

The following table shows various types of Avro, ORC, and Parquet column names, and rules for using the column names in Oracle column names in external tables.

Avro, ORC, or Parquet Name	CREATE TABLE Name	Oracle CATALOG	Valid SQL	Notes
part, Part, or PART	part, Part, PART	PART	`select part` `select Part` `select paRt` `select PART`	Oracle implicitly uppercases unquoted column names

Avro, ORC, or Parquet Name	CREATE TABLE Name	Oracle CATALOG	Valid SQL	Notes
Ord No	"Ord No"	Ord No	`select "Ord No"`	Double quotes are required when there are embedded blanks, which also preserves the character case
__index_key__	"__index_key__"	__index_key__	`select "__index_key__"`	Double quotes are required when there is a leading underscore, which also preserves the character case
6Way	"6Way"	6Way	`select "6Way"`	Double quotes are required when there is a leading numeric digit, which also preserves the character case
create, Create, or CREATE, and so on. (any case variation) partition, Partition, PARTITION, and so on (for an Oracle Reserved word)	"CREATE" "PARTITION"	CREATE PARTITION	`select "CREATE"` `select "PARTITION"`	Double quotes are required around Oracle SQL Reserved words. These are forced to uppercase, but must always be double-quoted when used anywhere in SQL
rowid, Rowid, ROWid, and so on (for ROWID see notes)	rowid		`select "rowid"` `select "Rowid"` `select "ROWid"` `select "rowid"`	For ROWID, any mixed or lower-case variation of ROWID preserves the case and must always be double-quoted and use the original case variations. Due to the inherent conflict with Oracle ROWID for the table, if you specify upper-case ROWID, it is automatically stored as lower-case "rowid" and must always be double-quoted when referenced.

Notes:

- In general a column name in an external table can be referenced without double quotes.
- Unless there is an embedded blank, a leading underscore ("_") or leading numeric digit ("0" through "9") in the column name, the original case of the column name is preserved, and it must always be referenced with double quotes and using the original case (upper, lower or mixed-case) of the Avro, ORC, or Parquet column name.
- After using `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` to create an external table with the format specified as `avro`, `orc`, or `parquet`, use the `DESCRIBE` command in SQL*Plus to view the table's column names.
- When Oracle SQL Reserved Words are used in Avro, ORC, or Parquet column names, they must always be double-quoted when referenced anywhere in SQL. See Oracle SQL Reserved Words for more information.

DBMS_CLOUD Exceptions

The following table describes exceptions for `DBMS_CLOUD`.

Exception	Code	Description
`reject_limit`	20003	The reject limit of an external table was reached.
`credential_not_exist`	20004	A credential object does not exist.
`table_not_exist`	20005	A table does not exist.
`unsupported_obj_store`	20006	An unsupported object store URI was provided.
`iden_too_long`	20008	An identifier is too long.
`invalid_format`	20009	A format argument is not valid.
`missing_credential`	20010	Mandatory credential object information was not specified.
`invalid_object_uri`	20011	An invalid object URI was provided.
`invalid_partitioning_clause`	20012	An partitioning clause is missing or was not provided.
`unsupported_feature`	20013	An unsupported feature was used that is not existent in the current database version.
`part_not_exist`	20014	A partition or subpartition does not exist, or a table is not a partitioned external table or hybrid partitioned table.
`invalid_table_name`	20016	An invalid table name was used.
`invalid_schema_name`	20017	An invalid schema name was used.
`invalid_dir_name`	20018	An invalid directory name was used.
`invalid_file_name`	20019	An invalid file name was used.
`invalid_cred_attribute`	20020	Invalid credential attributes were specified.
`table_exist`	20021	A table already exists.
`credential_exist`	20022	A credential object already exists.
`invalid_req_method`	20023	A request method is either too long or invalid.
`invalid_req_header`	20024	An invalid request header was specified.
`file_not_exist`	20025	A file does not exist.
`invalid_response`	20026	An HTTP response was not valid.

Exception	Code	Description
invalid_operation	20027	An invalid task class or ID was specified.
invalid_user_name	20028	An invalid username was specified.

DBMS_CLOUD_ADMIN Package

The DBMS_CLOUD_ADMIN package provides administrative routines for configuring a database.

- Summary of DBMS_CLOUD_ADMIN Subprograms
  This section covers the DBMS_CLOUD_ADMIN subprograms provided with Autonomous Database.
- DBMS_CLOUD_ADMIN Exceptions
  The following table describes exceptions for DBMS_CLOUD_ADMIN.

Summary of DBMS_CLOUD_ADMIN Subprograms

This section covers the DBMS_CLOUD_ADMIN subprograms provided with Autonomous Database.

Subprogram	Description
ATTACH_FILE_SYSTEM Procedure	This procedure attaches a file system in a directory on your database.
CANCEL_WORKLOAD_CAPTURE Procedure	This procedure cancels the current workload capture.
CREATE_DATABASE_LINK Procedure	This procedure creates a database link to a target database. There are options to create a database link to another Autonomous Database instance, to an Oracle Database that is not an Autonomous Database, or to a non-Oracle Database using Oracle-managed heterogeneous connectivity.
DETACH_FILE_SYSTEM Procedure	This procedure detaches a file system from a directory on your database.
DISABLE_APP_CONT Procedure	This procedure disables database application continuity for the session associated with the specified service name in Autonomous Database.
DISABLE_EXTERNAL_AUTHENTICATION Procedure	This procedure disables external authentication for the Autonomous Database instance.
DISABLE_OPERATOR_ACCESS Procedure	Immediately revokes Cloud Operator access on the Autonomous Database Database instance.
DISABLE_PRINCIPAL_AUTH Procedure	This procedure revokes principal based authentication for the specified provider and applies to the ADMIN user or to the specified user.
DISABLE_RESOURCE_PRINCIPAL Procedure	This procedure disables resource principal credential and creates the credential OCI$RESOURCE_PRINCIPAL. With a user name specified, other than ADMIN, the procedure grants the specified schema access to the resource principal credential.
DROP_DATABASE_LINK Procedure	This procedure drops a database link.
ENABLE_APP_CONT Procedure	This procedure enables database application continuity for the session associated with the specified service name in Autonomous Database.

Subprogram	Description
ENABLE_AWS_ARN Procedure	This procedure enables a user to create AWS ARN credentials in Autonomous Database.
ENABLE_EXTERNAL_AUTHENTICATION Procedure	This procedure enables a user to logon to Autonomous Database using the specified external authentication scheme.
ENABLE_FEATURE Procedure	This procedure enables the specified feature on the Autonomous Database instance.
ENABLE_OPERATOR_ACCESS Procedure	Grants the Cloud Operator access to an Autonomous Database instance for a specified number of hours.
ENABLE_PRINCIPAL_AUTH Procedure	This procedure enables principal authentication for the specified provider and applies to the ADMIN user or the specified user.
ENABLE_RESOURCE_PRINCIPAL Procedure	This procedure enables resource principal credential and creates the credential `OCI$RESOURCE_PRINCIPAL`. With a user name specified, other than ADMIN, the procedure grants the specified schema access to the resource principal credential.
FINISH_WORKLOAD_CAPTURE Procedure	This procedure stops the workload capture and uploads capture files to object storage.
PREPARE_REPLAY Procedure	This procedure prepares replay for the refreshable clone.
PURGE_FLASHBACK_ARCHIVE Procedure	This procedure purges historical data from the Flashback Data Archive.
REPLAY_WORKLOAD Procedure	This procedure is overloaded. It initiates the workload replay.
SET_FLASHBACK_ARCHIVE_RETENTION Procedure	This procedure enables ADMIN users to modify the retention period for Flashback Time Travel `flashback_archive`.
START_WORKLOAD_CAPTURE Procedure	This procedure initiates a workload capture.

- ATTACH_FILE_SYSTEM Procedure
  This procedure attaches a file system in the database.
- CANCEL_WORKLOAD_CAPTURE Procedure
  This procedure cancels any ongoing workload capture on the database.
- CREATE_DATABASE_LINK Procedure
- DETACH_FILE_SYSTEM Procedure
  This procedure detaches a file system from the database.
- DISABLE_APP_CONT Procedure
  This procedure disables database application continuity for the session associated with the specified service name in Autonomous Database.
- DISABLE_EXTERNAL_AUTHENTICATION Procedure
  Disables user authentication with external authentication schemes for the database.
- DISABLE_FEATURE Procedure
  This procedure disables the specified feature on the Autonomous Database instance.
- DISABLE_OPERATOR_ACCESS Procedure
  This procedure immediately revokes Cloud Operator access on the Autonomous Database instance.

- DISABLE_PRINCIPAL_AUTH Procedure
  This procedure revokes principal based authentication for a specified provider on Autonomous Database and applies to the ADMIN user or to the specified user.
- DISABLE_RESOURCE_PRINCIPAL Procedure
  Disable resource principal credential for the database or for the specified schema.
- DROP_DATABASE_LINK Procedure
  This procedure drops a database link.
- ENABLE_APP_CONT Procedure
  This procedure enables database application continuity for the session associated with the specified service name in Autonomous Database.
- ENABLE_AWS_ARN Procedure
  This procedure enables an Autonomous Database instance to use Amazon Resource Names (ARNs) to access AWS resources.
- ENABLE_EXTERNAL_AUTHENTICATION Procedure
  Enable users to login to the database with external authentication schemes.
- ENABLE_FEATURE Procedure
  This procedure enables the specified feature on the Autonomous Database instance.
- ENABLE_OPERATOR_ACCESS Procedure
  Oracle Cloud Operations does not access your Autonomous Database instance and access is disallowed by default. When access is required to troubleshoot or mitigate an issue, you can allow a cloud operator access to the database schemas for a limited time.
- ENABLE_PRINCIPAL_AUTH Procedure
  This procedure enables principal authentication on Autonomous Database for the specified provider and applies to the ADMIN user or the specified user.
- ENABLE_RESOURCE_PRINCIPAL Procedure
  Enable resource principal credential for the database or for the specified schema. This procedure creates the credential `OCI$RESOURCE_PRINCIPAL`.
- FINISH_WORKLOAD_CAPTURE Procedure
  This procedure finishes the current workload capture, stops any subsequent workload capture requests to the database, and uploads the capture files to Object Storage.
- PREPARE_REPLAY Procedure
  The `PREPARE_REPLAY` procedure prepares the refreshable clone for a replay.
- PURGE_FLASHBACK_ARCHIVE Procedure
  This procedure enables ADMIN users to purge historical data from Flashback Data Archive. You can either purge all historical data from Flashback Data Archive `flashback_archive` or selective data based on timestamps or System Change Number.
- REPLAY_WORKLOAD Procedure
  This procedure initiates a workload replay on your Autonomous Database instance. The overloaded form enables you to replay the capture files from an Autonomous Database instance, on-premises database, or other cloud service databases.
- SET_FLASHBACK_ARCHIVE_RETENTION Procedure
  This procedure allows ADMIN users to modify the retention period for Flashback Data Archive `flashback_archive`.
- START_WORKLOAD_CAPTURE Procedure
  This procedure initiates a workload capture on your Autonomous Database instance.

ATTACH_FILE_SYSTEM Procedure

This procedure attaches a file system in the database.

The `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` procedure attaches a file system in your database and stores information about the file system in the `DBA_CLOUD_FILE_SYSTEMS` view.

Syntax

```
DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM (
    file_system_name         IN VARCHAR2,
    file_system_location     IN VARCHAR2,
    directory_name           IN VARCHAR2,
    description              IN VARCHAR2 DEFAULT NULL,
    params                   IN CLOB DEFAULT NULL
);
```

Parameters

Parameter	Description
`file_system_name`	Specifies the name of the file system. This parameter is mandatory.
`file_system_location`	Specifies the location of the file system. The value you supply with `file_system_location` consists of a Fully Qualified Domain Name (FQDN) and a file path in the form: *FQDN*:*file_path*. For example: • FQDN: `myhost.sub000445.myvcn.oraclevcn.com` For Oracle Cloud Infrastructure File Storage set the FQDN in **Show Advanced Options** when you create a file system. See Creating File Systems for more information. • File Path: `/results` This parameter is mandatory.
`directory_name`	Specifies the directory name for the attached file system. The directory must exist. This parameter is mandatory.
`description`	(Optional) Provides a description of the task.
`params`	A JSON string that provides an additional parameter for the file system. • `nfs_version`: Specifies the NFS version to use when NFS is attached (NFSv3 or NFSv4). Valid values: 3, 4. Default value: 3

Examples:

Attach to an NFSv3 file system:

```
BEGIN
   DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM (
    file_system_name      => 'FSS',
    file_system_location  => 'myhost.sub000445.myvcn.oraclevcn.com:/results',
    directory_name        => 'FSS_DIR',
    description           => 'Source NFS for sales data'
```

```
  );
END;
/
```

Attach to an NFSv4 file system:

```
BEGIN
   DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM (
    file_system_name      => 'FSS',
    file_system_location  => 'myhost.sub000445.myvcn.oraclevcn.com:/results',
    directory_name        => 'FSS_DIR',
    description           => 'Source NFS for sales data',
    params                => JSON_OBJECT('nfs_version' value 4)
);
END;
/
```

Usage Notes

- To run this procedure you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.
- You must have `WRITE` privilege on the directory object in the database to attach a file system using `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM`.
- The `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` procedure can only attach a private File Storage Service in databases with Private Endpoints enabled.

  See OCI File Storage Service and Configuring Network Access with Private Endpoints for more information.
- The `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` procedure looks up the Network File System hostname on the customer's virtual cloud network (VCN). The error `"ORA-20000: Mounting NFS fails"` is returned if the hostname specified in the location cannot be located.
- Oracle Cloud Infrastructure File Storage uses NFSv3 to share
- If you attach to non-Oracle Cloud Infrastructure File Storage systems, the procedure supports NFSv3 and NFSv4
- If you have an attached NFS server that uses NFSv3 and the NFS version is updated to NFSv4 in the NFS server, you must run `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` and then `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` (using the `params` parameter with `nfs_version` set to 4). This attaches NFS with the matching protocol so that Autonomous Database can access the NFSv4 Server. Without detaching and then reattaching, the NFS server will be inaccessible and you may see an error such as: `"Protocol not supported"`.

CANCEL_WORKLOAD_CAPTURE Procedure

This procedure cancels any ongoing workload capture on the database.

This procedure cancels the current workload capture and enables refresh on the refreshable clone.

Syntax

```
DBMS_CLOUD_ADMIN.CANCEL_WORKLOAD_CAPTURE;
```

Example

```
BEGIN
   DBMS_CLOUD_ADMIN.CANCEL_WORKLOAD_CAPTURE;
END;
/
```

Usage Note

- To run this procedure you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

CREATE_DATABASE_LINK Procedure

This procedure creates a database link to a target database in the schema calling the API.

The overloaded forms support the following:

- When you use the `gateway_params` parameter, this enables you to create a database link with Oracle-managed heterogeneous connectivity where the link is to a supported non-Oracle database.
- When you use the `rac_hostnames` parameter, this enables you to create a database link from an Autonomous Database on a private endpoint to a target Oracle RAC database. In this case, you use the `rac_hostnames` parameter to specify the host names of one or more individual nodes of the target Oracle RAC database.

Syntax

```
DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
       db_link_name          IN VARCHAR2,
       hostname              IN VARCHAR2,
       port                  IN NUMBER,
       service_name          IN VARCHAR2,
       ssl_server_cert_dn    IN VARCHAR2 DEFAULT,
       credential_name       IN VARCHAR2 DEFAULT,
       directory_name        IN VARCHAR2 DEFAULT,
       gateway_link          IN BOOLEAN DEFAULT,
       public_link           IN BOOLEAN DEFAULT,
       private_target        IN BOOLEAN DEFAULT
       gateway_params        IN CLOB DEFAULT);


DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
       db_link_name          IN VARCHAR2,
       rac_hostnames         IN CLOB,
       port                  IN NUMBER,
       service_name          IN VARCHAR2,
       ssl_server_cert_dn    IN VARCHAR2 DEFAULT,
       credential_name       IN VARCHAR2 DEFAULT,
       directory_name        IN VARCHAR2 DEFAULT,
       gateway_link          IN BOOLEAN DEFAULT,
       public_link           IN BOOLEAN DEFAULT,
       private_target        IN BOOLEAN DEFAULT);
```

Parameters

Parameter	Description
db_link_name	The name of the database link to create.
hostname	The hostname for the target database. Specifying localhost for hostname as is not allowed. When you specify a connection with Oracle-managed heterogeneous connectivity by supplying the gateway_params parameter, note the following: • When the db_type value is google_bigquery the hostname is not used and you can provide value such as example.com. • When the db_type value is snowflake the hostname is the Snowflake account identifier. To find your Snowflake account identifier, see Account Identifier Formats by Cloud Platform and Region. Use this parameter or rac_hostnames, do not use both.
rac_hostnames	Specifies hostnames for the target Oracle RAC database. The value is a JSON array that specifies one or more individual host names for the nodes of the target Oracle RAC database. Multiple host names can be passed in JSON, separated by a ",". For example: `'["sales1-svr1.domain", "sales1-svr2.domain",` `"sales1-svr3.domain"]'` When the target is an Oracle RAC database, use the rac_hostnames parameter to specify one or more hostnames with DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK. This allows you to take advantage of the high availability capabilities of Oracle RAC. Using an IP address, a SCAN IP, or a SCAN hostname in the rac_hostnames value is not supported. When you specify a list of host names in the rac_hostnames parameter, CREATE_DATABASE_LINK uses all of the specified host names as addresses in the connect string. If one of the specified hosts is not available on the target Oracle RAC database, Autonomous Database automatically attempts to connect using another host name from the list. Use this parameter or hostname, do not use both. Specifying localhost for a rac_hostname value is not allowed.

Parameter	Description
port	Specifies the port for the connections to the target database. When you specify a connection with Oracle-managed heterogeneous connectivity using the gateway_params parameter, set the port based on the db_type value: • awsredshift: use port 5439 • azure: use port 1433 • db2: use port 2500 for Db2 versions >= 11.5.6 • db2: use port 5000 for Db2 versions <= 11.5.5 • google_analytics: use port 443 • google_bigquery: use port 443 • hive: use port 433 • mongodb: use port 27017 • mysql: use port 3306 • mysql_community: use port 3306 • postgres: use port 5432 • salesforce: use port 19937 • servicenow: use port 443 • snowflake: use port 443 See Oracle-Managed Heterogeneous Connectivity Database Types and Ports for more information.
service_name	The service_name for the database to link to. For a target Autonomous Database, find the service name by one of the following methods: • Look in the tnsnames.ora file in the wallet.zip that you download from an Autonomous Database for your connection. • Click **Database connection** on the Oracle Cloud Infrastructure Console. In the **Connection Strings** area, each connection string includes a service_name entry with the connection string for the corresponding service. When both Mutual TLS (mTLS) and TLS connections are allowed, under **TLS authentication** select **TLS** to view the TNS names and connection strings for connections with TLS authentication. See View TNS Names and Connection Strings for an Autonomous Database Instance for more information. • Query V$SERVICES view. For example: `SELECT name FROM V$SERVICES;` When you specify a connection with Oracle-managed heterogeneous connectivity using the gateway_params parameter, the service_name is the database name of the non-Oracle database.

Parameter	Description
ssl_server_cert_dn	The DN value found in the server certificate. Oracle-managed heterogeneous connectivity is preconfigured with a wallet that contains most of the common trusted root and intermediate SSL certificates. The ssl_server_cert_dn must be NULL when you supply the gateway_params parameter or do not include the ssl_server_cert_dn parameter (the default value is NULL). **Public Endpoint Link to an Autonomous Database Target without a Wallet:** To connect to an Autonomous Database target on a public endpoint without a wallet (TLS): • The directory_name parameter must be NULL. • The ssl_server_cert_dn parameter must be NULL or do not include this parameter (the default value is NULL). **Private Endpoint Link without a Wallet:** To connect to an Oracle Database on a private endpoint without a wallet: • The target database must be on a private endpoint. • The directory_name parameter must be NULL. • The ssl_server_cert_dn parameter must be NULL or do not include this parameter (the default is NULL). • The private_target parameter must be TRUE.
credential_name	The name of a stored credential created with DBMS_CLOUD.CREATE_CREDENTIAL. This is the credentials to access the target database.
directory_name	The directory for the cwallet.sso file. The default value for this parameter is 'data_pump_dir'. Oracle-managed heterogeneous connectivity is preconfigured with a wallet that contains most of the common trusted root and intermediate SSL certificates. The directory_name parameter is not required when you supply the gateway_params parameter. **Public Endpoint Link to an Autonomous Database Target without a Wallet:** To connect to an Autonomous Database on a public endpoint without a wallet (TLS): • The directory_name parameter must be NULL. • The ssl_server_cert_dn parameter must be NULL or do not include this parameter (the default value is NULL). In addition, to connect to an Autonomous Database with TCP, the ssl_server_cert_dn parameter must be NULL or do not include this parameter (the default value is NULL). **Private Endpoint Link without a Wallet:** To connect to a target Oracle Database on a private endpoint without a wallet: • The target database must be on a private endpoint. • The directory_name parameter must be NULL. • The ssl_server_cert_dn parameter must be NULL or do not include this parameter (the default value is NULL). • The private_target parameter must be TRUE.

Parameter	Description
gateway_link	Indicates if the database link is created to another Oracle Database or to an Oracle Database Gateway. If gateway_link is set to FALSE, this specifies a database link to another Autonomous Database or to another Oracle Database. If gateway_link is set to TRUE, this specifies a database link to a non-Oracle system. This creates a connect descriptor in the database link definition that specifies (HS=OK). When gateway_link is set to TRUE and gateway_params is NULL, this specifies a database link to a customer-managed Oracle gateway. The default value for this parameter is FALSE.
public_link	Indicates if the database link is created as a public database link. To run DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK with this parameter set to TRUE, the user invoking the procedure must have EXECUTE privilege on the credential associated with the public database link and must have the CREATE PUBLIC DATABASE LINK system privilege. The EXECUTE privilege on the credential can be granted either by the ADMIN user or by the credential owner. The default value for this parameter is FALSE.
private_target	When a database link accesses a hostname that needs to be resolved in a VCN DNS server, specify the private_target parameter with value TRUE. When private_target is TRUE, the hostname parameter must be a single hostname (on a private endpoint, using an IP address, a SCAN IP, or a SCAN hostname is not supported). The default value for this parameter is FALSE.

Parameter	Description
gateway_params	Specify the target database type for Oracle-managed heterogeneous connectivity to connect to non-Oracle databases. The db_type value is one of: • awsredshift • azure • db2 • google_analytics • google_bigquery * See Usage Notes for additional supported gateway_params when db_type is google_bigquery. • hive * See Usage Notes for additional supported gateway_params when db_type is hive. • mongodb • mysql • postgres • salesforce * See Usage Notes for additional supported gateway_params when db_type is salesforce. • servicenow * See Usage Notes for additional supported gateway_params when db_type is servicenow. • snowflake * See Usage Notes for additional supported gateway_params when db_type is snowflake. • NULL When gateway_params is NULL and gateway_link is set to TRUE, this specifies a database link to a customer-managed Oracle gateway. Specify the parameter with the json_object form. For example: gateway_params => json_object('db_type' value 'awsredshift') See Oracle-Managed Heterogeneous Connectivity Database Types and Ports for required port values for each database type. When gateway_params is NULL and private_target is TRUE, if directory_name is NULL, a TCP-based database link is created. When gateway_params is NULL and private_target is TRUE, if directory_name is NULL, a TCPS-based database link is created.

Usage Notes

- When you specify the gateway_params parameter, for some db_type values, additional gateway_params parameters are supported:

db_type	Additional gateway_params Values
google_analytics	When the db_type is google_analytics, the credential you specify must be a Google OAuth credential (gcp_oauth2) See CREATE_CREDENTIAL Procedure for more information.

db_type	Additional gateway_params Values
google_bigquery	When the db_type is google_bigquery, the credential you specify must be a Google OAuth credential (gcp_oauth2) See CREATE_CREDENTIAL Procedure for more information. When db_type is google_bigquery, the parameter project is valid. This parameter specifies the project name for google_bigquery and is required. The table name you specify when you use SELECT with Google BigQuery must be in quotes. For example: `SELECT * FROM "sales"@GOOGLE_BIGQUERY_LINK`
hive	When db_type is hive, the parameter http_path is valid. This parameter specifies the HttpPath value, if required, to connect to the Hive instance.
salesforce	When the db_type is salesforce, the parameter: security_token is valid. A security token is a case-sensitive alphanumeric code. Supplying a security_token value is required to access Salesforce. For example: `gateway_params => JSON_OBJECT(` `'db_type' value 'salesforce',` `'security_token' value 'security_token_value' )` See Reset Your Security Token for more information.

db_type	Additional gateway_params Values
servicenow	To connect to ServiceNow and get data you must supply the gateway parameters directory_name and file_name. These parameters specify a model file (REST config file) that maps the JSON response to the relational model. The model file specifies the endpoints, table mapping, and HTTP response code for processing the JSON response. See Model file syntax and Example Model file for more information. When you use gateway_params parameter with db_type servicenow, there are two supported options: – Basic Authentication: you must supply the gateway_params parameter db_type with the value 'servicenow', and supply the directory_name and file_name parameters along with username/password type credentials. – OAuth 2.0 Authentication: you must supply the gateway_params parameter db_type with the value 'servicenow', and the directory_name, file_name, and token_uri parameters, along with OAuth type credentials. The directory_name parameter specifies the directory with the ServiceNow REST config file. You could create this directory as follows: `create or replace directory servicenow_dir as 'SERVICENOW_DIR';` Obtain and download the ServiceNow REST config file to the specified directory. For example: `exec DBMS_CLOUD.get_object('servicenow_dir_cred', 'https://objectstorage.<...>/servicenow.rest','SERVICENOW_DIR');` Set the file_name value to the name of the REST config file you downloaded, "servicenow.rest". Then you can use the ServiceNow REST config file with either basic authentication or OAuth2.0. See HETEROGENEOUS_CONNECTIVITY_INFO View for samples.
snowflake	When the db_type is SNOWFLAKE, the parameters: role, schema, and warehouse are valid. These values specify a different schema, role, or warehouse value, other than the default. For example: `gateway_params => JSON_OBJECT( 'db_type' value 'snowflake', 'role' value 'ADMIN', 'schema' value 'PUBLIC', 'warehouse' value 'TEST' )`

- When you use the private_target parameter, note that database links from an Autonomous Database to a database service that is on a private endpoint are only supported in commercial regions and US Government regions.

  This feature is enabled by default in all commercial regions.

  This feature is enabled by default in US Government regions for newly provisioned databases.

For existing US Government databases on a private endpoint, if you want to create database links from an Autonomous Database to a target in a US Government region, please file a Service Request at Oracle Cloud Support and request to enable the private endpoint in government regions database linking feature.

US Government regions include the following:

- Oracle Cloud Infrastructure US Government Cloud with FedRAMP Authorization
- Oracle Cloud Infrastructure US Federal Cloud with DISA Impact Level 5 Authorization

- When connecting to a non-Oracle database, database linking is only supported if the target database is accessible through a public IP or a public hostname. See Create Database Links to Non-Oracle Databases for more information.
- To run `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` with a user other than ADMIN, you need to grant `EXECUTE` and `CREATE DATABASE LINK` privileges to that user. For example, run the following command as ADMIN to grant privileges to `adb_user`:

```
GRANT EXECUTE ON DBMS_CLOUD_ADMIN TO adb_user;
GRANT CREATE DATABASE LINK TO adb_user;
```

  In addition, when you create a Database Link in a schema other than the ADMIN schema, for example in a schema named `adb_user`, the `adb_user` schema must own the credential you use with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.
- Only one wallet file is valid per directory specified with the `directory_name` parameter. You can only upload one `cwallet.sso` at a time to the directory you choose for wallet files. This means with a `cwallet.sso` in a directory, you can only create database links to the databases for which the wallet in that directory is valid. To use multiple `cwallet.sso` files with database links you need to create additional directories and put each `cwallet.sso` in a different directory.

  See Create Directory in Autonomous Database for information on creating directories.
- To create a database link to an Autonomous Database, set `GLOBAL_NAMES` to `FALSE` on the source database (non-Autonomous Database).

```
SQL> ALTER SYSTEM SET GLOBAL_NAMES = FALSE;

System altered.

SQL> SHOW PARAMETER GLOBAL_NAMES
NAME                    TYPE        VALUE
----------------------  ----------- -----------
global_names            boolean     FALSE
```

- When the `private_target` parameter is `TRUE`, the `hostname` parameter specifies a private host inside the VCN.

Examples

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
     credential_name => 'DB_LINK_CRED',
     username => 'adb_user',
     password => 'password');
  DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
```

```
     db_link_name => 'SALESLINK',
     hostname => 'adb.eu-frankfurt-1.oraclecloud.com',
     port => '1522',
     service_name => 'example_medium.adb.example.oraclecloud.com',
     ssl_server_cert_dn => 'CN=adb.example.oraclecloud.com,OU=Oracle BMCS
FRANKFURT,O=Oracle Corporation,L=Redwood City,ST=California,C=US',
     credential_name => 'DB_LINK_CRED');
END;
/


BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'AWS_REDSHIFT_LINK_CRED',
    username => 'NICK',
    password => 'password'
  );
DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
          db_link_name => 'AWSREDSHIFT_LINK',
          hostname => 'example.com',
          port => '5439',
          service_name => 'example_service_name',
          ssl_server_cert_dn => NULL,
          credential_name => 'AWS_REDSHIFT_LINK_CRED',
          gateway_params => JSON_OBJECT('db_type'  value 'awsredshift'));
END;
/


BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'PRIVATE_ENDPOINT_CRED',
    username => 'db_user',
    password => 'password'
  );
DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
          db_link_name => 'PRIVATE_ENDPOINT_DB_LINK',
          hostname => 'exampleHostname',
          port => '1521',
          service_name => 'exampleServiceName',
          credential_name => 'PRIVATE_ENDPOINT_CRED',
          ssl_server_cert_dn => NULL,
          directory_name => NULL,
          private_target => TRUE);
END;
/


BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'GOOGLE_BIGQUERY_CRED',
    params => JSON_OBJECT( 'gcp_oauth2' value JSON_OBJECT(
          'client_id' value 'client_id',
          'client_secret' value 'client_secret',
          'refresh_token' value 'refresh_token' )));
```

```
DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
          db_link_name => 'GOOGLE_BIGQUERY_LINK',
          hostname => 'example.com',
          port => '443',
          service_name => 'example_service_name',
          credential_name => 'GOOGLE_BIGQUERY_CRED',
          gateway_params => JSON_OBJECT(
                    'db_type' value 'google_bigquery',
                    'project' value 'project_name1' ));
END;
/
```

The table name you specify when you use `SELECT` with Google BigQuery or Google Analytics must be in quotes. For example:

```
SELECT * FROM "sales"@GOOGLE_BIGQUERY_LINK
```

Use the `rac_hostnames` parameter with a target Oracle RAC database on a private endpoint.

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
     credential_name => 'DB_LINK_CRED1',
     username => 'adb_user',
     password => 'password');
      DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
     db_link_name => 'SALESLINK',
     rac_hostnames => '["sales1-svr1.example.adb.us-
ashburn-1.oraclecloud.com",
                        "sales1-svr2.example.adb.us-
ashburn-1.oraclecloud.com",
                        "sales1-svr3.example.adb.us-
ashburn-1.oraclecloud.com"]',
     port => '1522',
     service_name => 'example_high.adb.oraclecloud.com',
     ssl_server_cert_dn => 'CN=adb.example.oraclecloud.com,OU=Oracle BMCS
FRANKFURT,O=Oracle Corporation,L=Redwood City,ST=California,C=US',
     credential_name => 'DB_LINK_CRED1',
     directory_name => 'EXAMPLE_WALLET_DIR',
     private_target => TRUE);
END;
/
```

DETACH_FILE_SYSTEM Procedure

This procedure detaches a file system from the database.

The `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` procedure detaches a file system from your database. In addition to that, the `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` procedure also removes the information about the file system from the `DBA_CLOUD_FILE_SYSTEMS` view.

Syntax

```
DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM(
    file_system_name          IN VARCHAR2
);
```

Parameters

Parameter	Description
file_system_name	Specifies the name of the file system. This parameter is mandatory.

Example:

```
BEGIN
   DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM (
    file_system_name       => 'FSS'
  );
END;
/
```

Usage Notes

- To run this procedure, you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.
- You must have the `WRITE` privilege on the directory object in the database, to detach a file system from a directory using the `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` procedure.
- The `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` procedure can only detach a private File Storage Service in databases with Private Endpoints enabled.

  See OCI File Storage Service and Configuring Network Access with Private Endpoints for more information.
- The `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` procedure looks up the Network File System hostname on the customer's virtual cloud network (VCN). The error `"ORA-20000: Mounting NFS fails"` is returned if the hostname specified in the location cannot be located.

DISABLE_APP_CONT Procedure

This procedure disables database application continuity for the session associated with the specified service name in Autonomous Database.

Syntax

```
DBMS_CLOUD_ADMIN.DISABLE_APP_CONT(
    service_name       IN VARCHAR2);
```

Parameters

Parameter	Description
service_name	The service_name for the Autonomous Database service. To find service names: • Look in the tnsnames.ora file in the wallet.zip that you download from an Autonomous Database for your connection. • Click **Database connection** on the Oracle Cloud Infrastructure Console. In the **Connection strings** area, each connection string includes a service_name entry that contains the connection string for the corresponding service. When both Mutual TLS (mTLS) and TLS connections are allowed, under **TLS authentication** select **TLS** to view the TNS names and connection strings for connections with TLS authentication. See View TNS Names and Connection Strings for an Autonomous Database Instance for more information. • Query V$SERVICES view. For example: `SELECT name FROM V$SERVICES;`

Usage Notes

See Overview of Application Continuity for more information on Application Continuity.

Example

```
BEGIN
    DBMS_CLOUD_ADMIN.DISABLE_APP_CONT(
        service_name => 'nv123abc1_adb1_high.adb.oraclecloud.com' );
END;
/
```

Verify the value as follows:

```
SELECT name, failover_type FROM DBA_SERVICES;

NAME                                                     FAILOVER_TYPE
-------------------------------------------------------- --------------
nv123abc1_adb1_high.adb.oraclecloud.com
```

DISABLE_EXTERNAL_AUTHENTICATION Procedure

Disables user authentication with external authentication schemes for the database.

Syntax

```
DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
```

Exceptions

Exception	Error	Description
invalid_ext_auth	ORA-20004	See the accompanying message for a detailed explanation.

Example

```
BEGIN
   DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
END;
/

PL/SQL procedure successfully completed.
```

DISABLE_FEATURE Procedure

This procedure disables the specified feature on the Autonomous Database instance.

Syntax

```
DBMS_CLOUD_ADMIN.DISABLE_FEATURE(
    feature_name    IN  VARCHAR2);
```

Parameters

Parameter	Description
feature_name	Specifies the feature type to be disabled. Supported values are: • 'ORAMTS': Disable OraMTS feature. • 'AUTO_DST_UPGRADE': Disable AUTO DST feature. • 'AUTO_DST_UPGRADE_EXCL_DATA': Disable AUTO DST EXCL DATA feature. • 'OWM': Disable Oracle Workspace Manager. • 'WORKLOAD_AUTO_REPLAY': Disable workload auto replay feature. This parameter is mandatory.

Examples

```
BEGIN
DBMS_CLOUD_ADMIN.DISABLE_FEATURE(
    feature_name => 'ORAMTS');
END;
/


BEGIN
DBMS_CLOUD_ADMIN.DISABLE_FEATURE(
    feature_name => 'AUTO_DST_UPGRADE');
```

```
END;
/


BEGIN
DBMS_CLOUD_ADMIN.DISABLE_FEATURE(
    feature_name => 'AUTO_DST_UPGRADE_EXCL_DATA');
END;
/


BEGIN
DBMS_CLOUD_ADMIN.DISABLE_FEATURE(
    feature_name => 'OWM');
END;
/


BEGIN
DBMS_CLOUD_ADMIN.DISABLE_FEATURE(
    feature_name => 'WORKLOAD_AUTO_REPLAY');
END;
/
```

Usage Notes

- To disable the `OraMTS`, `AUTO_DST_UPGRADE`, `AUTO_DST_UPGRADE_EXCL_DATA`, `OWM`, or `WORKLOAD_AUTO_REPLAY` features for your Autonomous Database instance, you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.
- When both `AUTO_DST_UPGRADE` and `AUTO_DST_UPGRADE_EXCL_DATA` are disabled, if new time zone versions are available the Autonomous Database instance does not upgrade to use the latest available time zone files.
- Query `dba_cloud_config` to verify that `AUTO_DST_UPGRADE` is disabled.

  ```
  SELECT param_name, param_value FROM dba_cloud_config WHERE
         LOWER(param_name) = 'auto_dst_upgrade';

  0 rows selected.
  ```

- Query `dba_cloud_config` to verify that `AUTO_DST_UPGRADE_EXCL_DATA` is disabled.

  ```
  SELECT param_name, param_value FROM dba_cloud_config WHERE
         LOWER(param_name) = 'auto_dst_upgrade_excl_data';

  0 rows selected.
  ```

DISABLE_OPERATOR_ACCESS Procedure

This procedure immediately revokes Cloud Operator access on the Autonomous Database instance.

Syntax

```
DBMS_CLOUD_ADMIN.DISABLE_OPERATOR_ACCESS;
```

Example

```
BEGIN
    DBMS_CLOUD_ADMIN.DISABLE_OPERATOR_ACCESS;
END;
/
```

DISABLE_PRINCIPAL_AUTH Procedure

This procedure revokes principal based authentication for a specified provider on Autonomous Database and applies to the ADMIN user or to the specified user.

Syntax

```
DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH(
       provider    IN VARCHAR2,
       username    IN VARCHAR2 DEFAULT 'ADMIN' );
```

Parameters

Parameter	Description
`provider`	Specifies the type of provider. Valid values: • `AWS` • `AZURE` • `GCP` • `OCI`
`username`	Specifier the user to disable principal based authentication for. A null value is valid for the `username`. If `username` is not specified, the procedure applies for the "`ADMIN`" user.

Usage Notes

- When the provider value is `AZURE` and the `username` is `ADMIN`, the procedure disables Azure service principal based authentication on Autonomous Database and deletes the Azure application on the Autonomous Database instance.
- When the provider value is `AZURE` and the `username` is a user other than the `ADMIN` user, the procedure revokes the privileges from the specified user. The `ADMIN` user and other users that are enabled to use the Azure service principal can continue to use `ADMIN.AZURE$PA` and the application that is created for the Autonomous Database instance remains on the instance.

Examples

```
BEGIN
     DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH(
```

```
       provider => 'AZURE',
       username => 'SCOTT');
END;
/


BEGIN
     DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH(
       provider => 'GCP');
END;
/
```

DISABLE_RESOURCE_PRINCIPAL Procedure

Disable resource principal credential for the database or for the specified schema.

Syntax

```
DBMS_CLOUD_ADMIN.DISABLE_RESOURCE_PRINCIPAL(
   username          IN VARCHAR2);
```

Parameter

Parameter	Description
`username`	Specifies an optional user name. The name of the database schema to remove resource principal access. If you do not supply a `username`, the `username` is set to ADMIN and the command removes the `OCI$RESOURCE_PRINCIPAL` credential.

Exceptions

Exception	Error	Description
`resource principal is already disabled`	`ORA-20031`	If you attempt to disable the resource principal when it is already disabled.

Usage Notes

- Resource principal is not available with refreshable clones.
- You must set up a dynamic group and policies for the dynamic group before you call `DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL`.

  See the following for more information on creating policies, creating a dynamic group, and creating rules:

  - Use Resource Principal to Access Oracle Cloud Infrastructure Resources
  - Managing Dynamic Groups
  - Managing Policies
- Verify that a resource principal credential is enabled by querying one of the views: DBA_CREDENTIALS or ALL_TAB_PRIVS.

For example, as the ADMIN user query the view DBA_CREDENTIALS:

```
SELECT owner, credential_name FROM dba_credentials
        WHERE credential_name = 'OCI$RESOURCE_PRINCIPAL' AND owner =
'ADMIN';

OWNER  CREDENTIAL_NAME
-----  ----------------------
ADMIN  OCI$RESOURCE_PRINCIPAL
```

For example, as a non-ADMIN user query the view ALL_TAB_PRIVS:

```
SELECT grantee, table_name, grantor, FROM ALL_TAB_PRIVS
          WHERE  grantee = 'ADB_USER';

GRANTEE   TABLE_NAME GRANTOR
--------- -----------------------------------
ADB_USER  OCI$RESOURCE_PRINCIPAL ADMIN
```

Example

```
EXEC DBMS_CLOUD_ADMIN.DISABLE_RESOURCE_PRINCIPAL();

PL/SQL procedure successfully completed.

SQL> select owner, credential_name from dba_credentials where credential_name
= 'OCI$RESOURCE_PRINCIPAL';

No rows selected.
```

DROP_DATABASE_LINK Procedure

This procedure drops a database link.

Syntax

```
DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK(
        db_link_name      IN VARCHAR2,
        public_link       IN BOOLEAN DEFAULT);
```

Parameters

Parameter	Description
db_link_name	The name of the database link to drop.
public_link	To run DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK with public_link set to TRUE, you must have the DROP PUBLIC DATABASE LINK system privilege. The default value for this parameter is FALSE.

Example

```
BEGIN
    DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK(
        db_link_name => 'SALESLINK' );
END;
/
```

Usage Notes

After you are done using a database link and you run `DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK`, to ensure security of your Oracle database remove any stored wallet files. For example:

- Remove the wallet file in Object Store.
- Use `DBMS_CLOUD.DELETE_FILE` to remove the wallet file from the `data_pump_dir` directory or from the user defined directory where the wallet file was uploaded.

ENABLE_APP_CONT Procedure

This procedure enables database application continuity for the session associated with the specified service name in Autonomous Database.

Syntax

```
DBMS_CLOUD_ADMIN.ENABLE_APP_CONT(
    service_name      IN VARCHAR2);
```

Parameters

Parameter	Description
`service_name`	The `service_name` for the Autonomous Database service. To find service names: • Look in the `tnsnames.ora` file in the `wallet.zip` that you download from an Autonomous Database for your connection. • Click **Database connection** on the Oracle Cloud Infrastructure Console. In the **Connection strings** area, each connection string includes a `service_name` entry that contains the connection string for the corresponding service. When both Mutual TLS (mTLS) and TLS connections are allowed, under **TLS authentication** select **TLS** to view the TNS names and connection strings for connections with TLS authentication. See View TNS Names and Connection Strings for an Autonomous Database Instance for more information. • Query `V$SERVICES` view. For example: `SELECT name FROM V$SERVICES;`

Usage Notes

See Overview of Application Continuity for more information on Application Continuity.

Example

```
BEGIN
    DBMS_CLOUD_ADMIN.ENABLE_APP_CONT(
        service_name => 'nvthp2ht_adb1_high.adb.oraclecloud.com'
    );
END;
/
```

Verify the value as follows:

```
SELECT name, failover_type FROM DBA_SERVICES;

NAME                                                  FAILOVER_TYPE
----------------------------------------------------- -------------
nvthp2ht_adb1_high.adb.oraclecloud.com                TRANSACTION
```

ENABLE_AWS_ARN Procedure

This procedure enables an Autonomous Database instance to use Amazon Resource Names (ARNs) to access AWS resources.

Syntax

```
DBMS_CLOUD_ADMIN.ENABLE_AWS_ARN(
       username     IN VARCHAR2 DEFAULT NULL,
       grant_option IN BOOLEAN DEFAULT FALSE);
```

Parameters

Parameter	Description
`username`	Name of the user to enable to use Amazon Resource Names (ARNs). A null value is valid for the `username`. If `username` is not specified, the procedure applies for the "`ADMIN`" user.
`grant_option`	When `username` is supplied, if `grant_option` is `TRUE` the specified `username` can enable Amazon Resource Names (ARNs) usage for other users.

Example

```
BEGIN
    DBMS_CLOUD_ADMIN.ENABLE_AWS_ARN(
        username => 'adb_user');
END;
/
```

Usage Note

- You must be the ADMIN user to run the procedure `DBMS_CLOUD_ADMIN.ENABLE_AWS_ARN`.

See Use Amazon Resource Names (ARNs) to Access AWS Resources for more information.

ENABLE_EXTERNAL_AUTHENTICATION Procedure

Enable users to login to the database with external authentication schemes.

Syntax

```
DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
   type         IN VARCHAR2,
   force        IN BOOLEAN DEFAULT FALSE,
   params       IN CLOB DEFAULT NULL
);
```

Parameter

Parameter	Description
`type`	Specifies the external authentication type. Valid values: or . • `'OCI_IAM'` • `'AZURE_AD'` • `'CMU'` • `'KERBEROS'`
`force`	(Optional) Override a currently enabled external authentication scheme. Valid values are `TRUE` or `FALSE`. The default value is `FALSE`.

Parameter	Description
params	A JSON string that provides additional parameters for the external authentication. CMU parameters: • location_uri: specifies the cloud storage URI for the bucket where files required for CMU are stored. If you specify location_uri there is a fixed name directory object CMU_WALLET_DIR created in the database at the path 'cmu_wallet' to save the CMU configuration files. In this case, you do not need to supply the directory_name parameter. • credential_name: specifies the credentials that are used to download the CMU configuration files from the Object Store to the directory object. Default value is NULL which allows you to provide a Public, Preauthenticated, or pre-signed URL for Object Store bucket or subfolder. • directory_name: specifies the directory name where configuration files required for CMU are stored. If directory_name is supplied, you are expected to copy the CMU configuration files dsi.ora and cwallet.sso to this directory object. KERBEROS parameters: • location_uri: specifies the cloud storage URI for the bucket where the files required for Kerberos are stored. If you specify location_uri there is a fixed name directory object KERBEROS_DIR created in the database to save the Kerberos configuration files. In this case, you do not need to supply the directory_name parameter. • credential_name: specifies the credential that are used to download Kerberos configuration files from the Object Store location to the directory object. Default value is NULL which allows you to provide a Public, Preauthenticated, or pre-signed URL for Object Store bucket or subfolder. • directory_name: specifies the directory name where files required for Kerberos are stored. If directory_name is supplied, you are expected to copy the Kerberos configuration files to this directory object. • kerberos_service_name: specifies a name to use as the Kerberos service name. This parameter is optional. Default value: When not specified, the kerberos_service_name value is set to the Autonomous Database instance's GUID. AZURE_AD parameters: • tenant_id: Tenant ID of the Azure Account. Tenant Id specifies the Autonomous Database instance's Azure AD application registration. • application_id: Azure Application ID created in Azure AD to assign roles/schema mappings for external authentication in the Autonomous Database instance. • application_id_uri: Unique URI assigned to the Azure Application. This it the identifier for the Autonomous Database instance. The name must be domain qualified (this supports cross tenancy resource access). The maximum length for this parameter is 256 characters.

Exceptions

Exception	Error	Description
invalid_ext_auth	ORA-20004	See the accompanying message for a detailed explanation.

Usage Notes

- With `type OCI_IAM`, if the resource principal is not enabled on the Autonomous Database instance, this routine enables resource principal with `DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL`.
- This procedure sets the system parameters `IDENTITY_PROVIDER_TYPE` and `IDENTITY_PROVIDER_CONFIG` to required users to access the instance with Oracle Cloud Infrastructure Identity and Access Management authentication and authorization.

Examples

Enable `OCI_IAM` Authentication

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
     type => 'OCI_IAM',
     force=> TRUE );
END;
/

PL/SQL procedure successfully completed.
```

Enable `CMU` Authentication for Microsoft Active Directory

You pass in a directory name that contains the CMU configuration files through `params` JSON argument.

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
     type => 'CMU',
     force => TRUE,
     params => JSON_OBJECT('directory_name' value 'CMU_DIR'); // CMU_DIR
directory object already exists
END;
/

PL/SQL procedure successfully completed.
```

You pass in a location URI pointing to an Object Storage location that contains CMU configuration files through `params` JSON argument.

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
       type     => 'CMU',
       params   => JSON_OBJECT('location_uri' value 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
                               'credential_name' value 'my_credential_name')
   );
```

```
END;
/

PL/SQL procedure successfully completed.
```

Enable Azure AD Authentication

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
     type => 'AZURE_AD',
     force => TRUE,
     params   => JSON_OBJECT( 'tenant_id' VALUE '....',
                              'application_id' VALUE '...',
                              'application_id_uri' VALUE '.....' ));
END;
/

PL/SQL procedure successfully completed.
```

Enable Kerberos Authentication

You pass in a directory name that contains Kerberos configuration files through `params` JSON argument.

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
     type => 'KERBEROS',
     force => TRUE,
     params => JSON_OBJECT('directory_name' value 'KERBEROS_DIR'); //
KERBEROS_DIR directory object already exists
END;
/

PL/SQL procedure successfully completed.
```

You pass in a location URI pointing to an Object Storage location that contains Kerberos configuration files through `params` JSON argument:

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
     type => 'KERBEROS',
     force => TRUE,
     params => JSON_OBJECT('location_uri' value 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
                           'credential_name' value 'my_credential_name');
END;
/
```

You pass in a service name with the `kerberos_service_name` in the `params` JSON argument:

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
     type => 'KERBEROS',
     force => TRUE,
     params => JSON_OBJECT('directory_name' value 'KERBEROS_DIR', //
KERBEROS_DIR directory object already exists
                           'kerberos_service_name' value 'oracle' ));
```

```
END;
/
```

After Kerberos is enabled on your Autonomous Database instance, use the following query to view the Kerberos service name:

```
SELECT SYS_CONTEXT('USERENV','KERBEROS_SERVICE_NAME') FROM DUAL;
```

ENABLE_FEATURE Procedure

This procedure enables the specified feature on the Autonomous Database instance.

Syntax

```
DBMS_CLOUD_ADMIN.ENABLE_FEATURE(
    feature_name     IN VARCHAR2,
    params           IN  CLOB    DEFAULT NULL);
```

Parameters

Parameter	Description
`feature_name`	Name of the feature to enable. The supported values are: • `'JAVAVM'`: Enable JAVAVM feature. • `'ORAMTS'`: Enable OraMTS feature. • `'AUTO_DST_UPGRADE'`: Enable AUTO DST feature. • `'AUTO_DST_UPGRADE_EXCL_DATA'`: Enable AUTO DST EXCL DATA feature. • `'OWM'`: Enable Oracle Workspace Manager. • `'WORKLOAD_AUTO_REPLAY'`: Enable the workload auto replay feature. This parameter is mandatory.

Parameter	Description
params	A JSON string that provides additional parameters for some features. For the OraMTS feature the params parameter is: • location_uri: the location_uri accepts a string value. The value specifies the HTTPS URL for the OraMTS server in a customer network. For the WORKLOAD_AUTO_REPLAY feature the params parameters are: • target_db_ocid: A string value. The value specifies the OCID of a target refreshable clone database on which the captured workload is replayed. The refreshable clone must have the **Early** patch level set. This parameter is mandatory. • capture_duration: A number value. The value specifies the duration in minutes for which the workload is captured on the production database. The value must be in the range between 1 and 720 minutes. This parameter is mandatory. • capture_day: A string value. The value specifies the day of the week the workload capture on the production database should begin. This parameter is optional. • capture_time: A value in the HH24:MM format. The value specifies the time of the day the workload capture on the production database should begin. This parameter is optional. By default the workload capture starts when you enable WORKLOAD_AUTO_REPLAY. If the optional capture_day and capture_time are specified, the capture and the replay happen at the specified timestamp. For example, if capture_day is Monday and capture_time is 15:00, the first capture happens at 3PM on the next Monday. The day of week and time are also used to schedule the later replay on the refreshable clone.

Example to Enable JAVAVM Feature:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_FEATURE (
        feature_name => 'JAVAVM' );
END;
/
```

Example to Enable Auto DST Feature:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_FEATURE (
        feature_name => 'AUTO_DST_UPGRADE' );
END;
/
```

Example to Enable Auto DST EXCL Data Feature:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_FEATURE (
        feature_name => 'AUTO_DST_UPGRADE_EXCL_DATA' );
END;
/
```

Example to Enable OraMTS Feature:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_FEATURE(
        feature_name => 'ORAMTS',
        params       => JSON_OBJECT('location_uri' VALUE 'https://
mymtsserver.mycorp.com')
   );
END;
/
```

Example to Enable OWM Feature:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_FEATURE(
        feature_name => 'OWM' );
END;
/
```

Example to Enable Workload Auto Replay Feature

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_FEATURE(
        feature_name => 'WORKLOAD_AUTO_REPLAY',
        params       => JSON_OBJECT('target_db_ocid' VALUE
'OCID1.TENANCY.REGION..ID1', 'capture_duration' VALUE 120, 'capture_day'
VALUE 'MONDAY', 'capture_time' VALUE '15:00'));
END;
/
```

Usage Notes

- You must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN` to run `DBMS_CLOUD_ADMIN.ENABLE_FEATURE`.
- After you run `DBMS_CLOUD_ADMIN.ENABLE_FEATURE` with `feature_name` value `'JAVAVM'`, you must restart the Autonomous Database instance to install Oracle Java.
- After you run `DBMS_CLOUD_ADMIN.ENABLE_FEATURE` with `feature_name` value `'OWM'`, you must restart the Autonomous Database instance to enable Oracle Workspace Manager. Oracle.
- To enable `AUTO_DST_UPGRADE`, `AUTO_DST_UPGRADE_EXCL_DATA`, `ORAMTS`, `JAVAVM`, `OWM`, or `WORKLOAD_AUTO_REPLAY` features for your database, you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.
- By default, both `AUTO_DST_UPGRADE` and `AUTO_DST_UPGRADE_EXCL_DATA` are disabled. You can enable one or the other of these options, but not both.
- After you enable `AUTO_DST_UPGRADE`, the next time you restart, or stop and then start the Autonomous Database instance, the instance upgrades to use the latest available time zone files. After `AUTO_DST_UPGRADE` is enabled, when new time zone files are available, the instance continues to upgrade to the latest available version on every subsequent restart or stop and start, until the feature is disabled.

Query `dba_cloud_config` to verify that `AUTO_DST_UPGRADE` is enabled.

```
SELECT param_name, param_value FROM dba_cloud_config WHERE
       LOWER(param_name) = 'auto_dst_upgrade';

PARAM_NAME       PARAM_VALUE
---------------- --------------
auto_dst_upgrade enabled
```

- After you enable `AUTO_DST_UPGRADE_EXCL_DATA`, the Autonomous Database instance upgrades to use the latest available time zone files. After this feature is enabled, every subsequent maintenance window upgrades the instance to use the latest available time zone version. This feature assures that the time zone files are upgraded for the database (enabling `AUTO_DST_UPGRADE_EXCL_DATA` does not update any affected rows).

  Query `dba_cloud_config` to verify that `AUTO_DST_UPGRADE_EXCL_DATA` is enabled.

```
SELECT param_name, param_value FROM dba_cloud_config WHERE
       LOWER(param_name) = 'auto_dst_upgrade_excl_data';

PARAM_NAME                    PARAM_VALUE
--------------------------    -----------
auto_dst_upgrade_excl_data    enabled
```

ENABLE_OPERATOR_ACCESS Procedure

Oracle Cloud Operations does not access your Autonomous Database instance and access is disallowed by default. When access is required to troubleshoot or mitigate an issue, you can allow a cloud operator access to the database schemas for a limited time.

Syntax

```
DBMS_CLOUD_ADMIN.ENABLE_OPERATOR_ACCESS(
    auth_duration    IN  NUMBER  DEFAULT 1
);
```

Parameters

Parameter	Description
`auth_duration`	Specifies the number of Hours for which Cloud Operator is granted access. Valid values: must be whole numbers in the range of 1 to 24. Default value: 1

Example

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_OPERATOR_ACCESS(
       auth_duration => 12 );
END;
/
```

Usage Notes

- `ORA-20000: Operator access is already enabled` indicates that operator access was already granted. In this case you have two options:
  - Wait for operator access to expire, and then grant operator access again with `DBMS_CLOUD_ADMIN.ENABLE_OPERATOR_ACCESS`
  - Explicitly disable operator access with DISABLE_OPERATOR_ACCESS Procedure.
- `DBMS_CLOUD_ADMIN.ENABLE_OPERATOR_ACCESS` allows access only to the Cloud Operator and does not enable access for any other user. All operations performed by the Cloud Operator are stored in the view `DBA_OPERATOR_ACCESS`. See View Oracle Cloud Infrastructure Operations Actions for more information.
- You allow a cloud operator to access the database schemas by running the procedure `DBMS_CLOUD_ADMIN.ENABLE_OPERATOR_ACCESS`. This means if you file a service request with Oracle Cloud Support and Oracle Cloud Operators need to access your database schemas, you must also enable operator access by running`DBMS_CLOUD_ADMIN.ENABLE_OPERATOR_ACCESS`.

ENABLE_PRINCIPAL_AUTH Procedure

This procedure enables principal authentication on Autonomous Database for the specified provider and applies to the ADMIN user or the specified user.

Syntax

```
DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
       provider    IN VARCHAR2,
       username    IN VARCHAR2 DEFAULT 'ADMIN',
       params      IN CLOB DEFAULT NULL);
```

Parameters

Parameter	Description
`provider`	Specifies the type of provider. Valid values: • `AWS`: Enable use of Amazon Resource Names (ARNs) • `AZURE`: Enable use of Azure Service Principal • `GCP`: Enable use of Google Service Account • `OCI`: Enable use of Resource Principal
`username`	Name of the user who has principal authentication usage enabled. A null value is valid for the `username`. If `username` is not specified, the procedure applies for the "`ADMIN`" user.

Parameter	Description
params	Specifies the configuration parameters. When the provider parameter is AWS, GCP, or OCI, params is not required. The default value is NULL. grant_option: This parameter is valid for all providers and is a Boolean value TRUE or FALSE. The default is FALSE. When TRUE and a username is specified, the specified user can use ENABLE_PRINCIPAL_AUTH to enable other users. **When the provider parameter is AWS, these options are also valid:** • aws_role_arn: Specifies the AWS role ARN. • external_id_type: Specifies the external ID type. Valid values are: "TENANT_OCID", "DATABASE_OCID", or "COMPARTMENT_OCID". The default value for external_id_type is "DATABASE_OCID". See Perform AWS Management Prerequisites to Use Amazon Resource Names (ARNs) for details on the values for external_id_type. **When the provider parameter is AZURE, this option is also valid:** • azure_tenantid: with the value of the Azure tenant ID.

Usage Notes

- When the provider parameter is AZURE, the params parameter must include the azure_tenantid in the following cases:
  - When DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH is called for the first time.
  - When DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH is called for the first time after DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH is called with the provider parameter AZURE and the username ADMIN.
- When the provider parameter is AWS:
  - After you enable ARN on the Autonomous Database instance by running DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH, the credential named AWS$ARN is available to use with any DBMS_CLOUD API that takes a credential as the input.

Examples

```
BEGIN
     DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
       provider => 'AZURE',
       username => 'SCOTT',
       params   => JSON_OBJECT('azure_tenantid' value 'azure_tenantid'));
END;
/


BEGIN
     DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
     provider => 'GCP',
     username => 'SCOTT',
     params => JSON_OBJECT(
         'grant_option' value 'TRUE' ));
```

```
END;
/


BEGIN
     DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
     provider => 'AWS',
     username => 'SCOTT',
     params => JSON_OBJECT(
         'aws_role_arn' value 'arn:aws:iam::123456:role/AWS_ROLE_ARN',
          'external_id_type' value 'TENANT_OCID'));
END;
/
```

ENABLE_RESOURCE_PRINCIPAL Procedure

Enable resource principal credential for the database or for the specified schema. This procedure creates the credential `OCI$RESOURCE_PRINCIPAL`.

Syntax

```
DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL(
   username         IN VARCHAR2,
   grant_option     IN BOOLEAN DEFAULT FALSE);
```

Parameter

Parameter	Description
`username`	Specifies an optional user name. The name of the database schema to be granted resource principal access. If you do not supply a `username`, the `username` is set to ADMIN.
`grant_option`	When `username` is supplied, if `grant_option` is `TRUE` the specified `username` can enable resource principal usage for other users.

Exceptions

Exception	Error	Description
`resource principal is already enabled`	`ORA-20031`	If you attempt to enable the resource principal when it is already enabled.

Usage Notes

- You must call `DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL` with the ADMIN `username` or with no arguments before you call `DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL` with a `username` for a database user schema.
- You must set up a dynamic group and policies for the dynamic group before you call `DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL`.

  See the following for more information on policies, creating a dynamic group, and creating rules:

  – Use Resource Principal to Access Oracle Cloud Infrastructure Resources

  - Managing Dynamic Groups
  - Managing Policies

- Enabling the resource principal with `DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL` is one-time operation. You do not need to enable the resource principal again, unless you run `DBMS_CLOUD_ADMIN.DISABLE_RESOURCE_PRINCIPAL` to disable the resource principal.
- Resource principal is not available with refreshable clones.

Example

```
EXEC DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL();

PL/SQL procedure successfully completed.

SQL> select owner, credential_name from dba_credentials where credential_name
= 'OCI$RESOURCE_PRINCIPAL';

OWNER    CREDENTIAL_NAME
-------  ---------------
ADMIN    OCI$RESOURCE_PRINCIPAL
```

FINISH_WORKLOAD_CAPTURE Procedure

This procedure finishes the current workload capture, stops any subsequent workload capture requests to the database, and uploads the capture files to Object Storage.

Example

```
BEGIN
    DBMS_CLOUD_ADMIN.FINISH_WORKLOAD_CAPTURE
END;
/
```

Usage Notes

- To run this procedure you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.
- When you pass the `duration` parameter to `START_WORKLOAD_CAPTURE`, the capture finishes when it reaches the specified time. However, if you call `FINISH_WORKLOAD_CAPTURE`, this stops the workload capture (possibly before the time specified with the duration parameter).

  You can query the `DBA_CAPTURE_REPLAY_STATUS` view to check the finish workload status. See DBA_CAPTURE_REPLAY_STATUS View for more information.

> **Note:**
>
> You must subscribe to the Information event `com.oraclecloud.databaseservice.autonomous.database.information` to be notified about the completion of `FINISH_WORKLOAD_CAPTURE` as well as the Object Storage link to download the capture file. This PAR URL is contained in the `captureDownloadURL` field of the event and is valid for 7 days from the date of generation. See Information Events on Autonomous Database for more information.

PREPARE_REPLAY Procedure

The `PREPARE_REPLAY` procedure prepares the refreshable clone for a replay.

Parameters

Parameter	Description
`capture_name`	Specifies the name of the workload capture. This parameter is mandatory.

Syntax

```
DBMS_CLOUD_ADMIN.PREPARE_REPLAY(
        capture_name IN VARCHAR2);
```

Example

```
BEGIN
    DBMS_CLOUD_ADMIN.PREPARE_REPLAY
      capture_name => 'cap_test1');
END;
/
```

This example prepares the refreshable clone to replay the workload indicated by the `capture_name` parameter, which involves bringing it up to the capture start time and then disconnecting it.

Usage Note

- To run this procedure you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

PURGE_FLASHBACK_ARCHIVE Procedure

This procedure enables ADMIN users to purge historical data from Flashback Data Archive. You can either purge all historical data from Flashback Data Archive `flashback_archive` or selective data based on timestamps or System Change Number.

Syntax

```
DBMS_CLOUD_ADMIN.PURGE_FLASHBACK_ARCHIVE(
   scope       IN VARCHAR2,
```

```
    before_scn IN INTEGER DEFAULT NULL,
    before_ts  IN TIMESTAMP DEFAULT NULL);
```

Parameter	Description
scope	This specifies the scope to remove data from the flashback data archive. • all implies PURGE ALL;before_scn and before_timestamp must both be NULL. • scn implies PURGE BEFORE SCN;before_scn must be non-NULL and before_timestamp must be NULL. • TIMESTAMP implies PURGE BEFORE timestamp;before_scn must be NULL and before_timestamp must be non-NULL.
before_scn	This specifies the system change number before which all the data is removed from the flashback archive.
before_timestamp	This specifies the timestamp before which all the data is removed from the flashback archive.

Example

```
BEGIN
    DBMS_CLOUD_ADMIN.PURGE_FLASHBACK_ARCHIVE(
           scope => 'ALL'); // Purge all historical data from Flashback Data
Archive flashback_archive
END;
/
```

REPLAY_WORKLOAD Procedure

This procedure initiates a workload replay on your Autonomous Database instance. The overloaded form enables you to replay the capture files from an Autonomous Database instance, on-premises database, or other cloud service databases.

Syntax

```
DBMS_CLOUD_ADMIN.REPLAY_WORKLOAD(
        capture_name                 IN VARCHAR2,
        replay_name                  IN VARCHAR2 DEFAULT NULL,
        capture_source_tenancy_ocid  IN VARCHAR2 DEFAULT NULL,
        capture_source_db_name       IN VARCHAR2 DEFAULT NULL);

DBMS_CLOUD_ADMIN.REPLAY_WORKLOAD(
        location_uri      IN VARCHAR2,
        credential_name   IN VARCHAR2 DEFAULT NULL,
        synchronization   IN BOOLEAN  DEFAULT TRUE,
        process_capture   IN BOOLEAN  DEFAULT TRUE);
```

Parameters

Parameter	Description
CAPTURE_NAME	Specifies the name of the workload capture. This parameter is mandatory.

Parameter	Description
REPLAY_NAME	Specifies the replay name. If you do not supply a REPLAY_NAME value, the REPLAY_NAME is auto-generated with the format *REPLAY_RANDOMNUMBER*, for example, REPLAY_1678329506.
CAPTURE_SOURCE_TENANCY_OCID	Specifies the source tenancy OCID of the workload capture. If you do not supply a CAPTURE_SOURCE_TENANCY_OCID value, the CAPTURE_SOURCE_TENANCY_OCID is set to NULL. This parameter is only mandatory when running the workload capture in a full clone.
CAPTURE_SOURCE_DB_NAME	Specifies the source database name of the workload capture If you do not supply a CAPTURE_SOURCE_DB_NAME value, the CAPTURE_SOURCE_DB_NAME is set to NULL. This parameter is only mandatory when running the workload capture in a full clone.
LOCATION_URI	Specifies URI that points to an Object Storage location that contains the captured files. This parameter is mandatory.
CREDENTIAL_NAME	Specifies the credential to access the object storage bucket. If you do not supply a credential_name value, the database's default credentials are used.
SYNCHRONIZATION	Specifies the synchronization method used during workload replay. • TRUE specifies that the synchronization is based on SCN. • FALSE specifies that the synchronization is based on TIME. If you do not supply a synchronization value, the synchronization is set to TRUE.
PROCESS_CAPTURE	Specifies whether or not you need to specify process_capture value. It can be set to FALSE only when you replay the same workload on the target database repeatedly. If you do not supply a process_capture value, the process_capture is set to TRUE.

Example to replay the workload from an on-premises database on an Autonomous Database instance:

```
BEGIN
  DBMS_CLOUD_ADMIN.REPLAY_WORKLOAD(
        location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
        credential_name => 'CRED_TEST',
        synchronization => TRUE,
        process_capture => TRUE);
END;
/
```

When you run this example, it:

- Downloads the capture files from the Object Storage location specified in location_uri and processes the capture files based on the process_capture parameter value.
- Replays the captured workload based on the synchronization parameter value.

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See Understanding Object Storage Namespaces for more information.

See Navigate to Oracle Cloud Infrastructure Object Storage and Create Bucket for more information on Object Storage.

See Upload Files to Your Oracle Cloud Infrastructure Object Store Bucket for more information on uploading files to Object Storage.

The `credential_name` you use in this step is the credentials for the Object Store.

You don't need to create a credential to access Oracle Cloud Infrastructure Object Store if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

Example to replay the workload from an Autonomous Database instance on another Autonomous Database:

```
BEGIN
  DBMS_CLOUD_ADMIN.REPLAY_WORKLOAD(
        capture_name => 'CAP_TEST1');
END;
/
```

When you run this example, it:

- Disconnects the current Autonomous Database instance.
- Downloads the capture files from the Object Storage.
- Replays the captured workload.
- Uploads replay report after a replay.

Usage Notes for Replaying the Workload from an On-Premises or Other Cloud Service Database on another Autonomous Database

- To run this procedure you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.
- Before you start replay, you should upload the `cap` and `capfiles` subdirectories, which contain the workload capture files, to the object storage location.

Usage Notes for Replaying the Workload from an Autonomous Database instance on another Autonomous Database

- To run this procedure you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.
- The replay files are automatically uploaded to the Object Store as a zip file.
- You can query the `DBA_CAPTURE_REPLAY_STATUS` view to check the workload replay status.

  See DBA_CAPTURE_REPLAY_STATUS View for more information.

> **Note:**
>
> You must subscribe to the Information event `com.oraclecloud.databaseservice.autonomous.database.information` to be notified about the start and completion of the `REPLAY_WORKLOAD` as well as the Object Storage link to download the replay reports. This PAR URL is contained in the `replayDownloadURL` field of the event and is valid for 7 days from the date of generation. See Information Events on Autonomous Database for more information.

SET_FLASHBACK_ARCHIVE_RETENTION Procedure

This procedure allows ADMIN users to modify the retention period for Flashback Data Archive `flashback_archive`.

Syntax

```
DBMS_CLOUD_ADMIN.SET_FLASHBACK_ARCHIVE_RETENTION (
    retention_days INTEGER);
```

Parameter	Description
`retention_days`	This specifies the length of time in days that the archived data should be retained for. The value of `retention_days` must be greater than 0.

Example

```
BEGIN
     DBMS_CLOUD_ADMIN.SET_FLASHBACK_ARCHIVE_RETENTION(
           retention_days => 90); // sets the retention time to 90 days
END;
/
```

START_WORKLOAD_CAPTURE Procedure

This procedure initiates a workload capture on your Autonomous Database instance.

Syntax

```
DBMS_CLOUD_ADMIN.START_WORKLOAD_CAPTURE(
        capture_name  IN VARCHAR2,
        duration      IN NUMBER   DEFAULT NULL);
```

Parameters

Parameter	Description
`capture_name`	Specifies the name of the workload capture. This parameter is mandatory.
`duration`	Specifies the duration in minutes for which you want to run the workload capture. • If you do not supply a `duration` value, the `duration` is set to NULL. • If set to NULL, the workload will continue until you run the `FINISH_WORKLOAD_CAPTURE` procedure.

Example

```
BEGIN
  DBMS_CLOUD_ADMIN.START_WORKLOAD_CAPTURE(
        capture_name => 'test');
END;
/
```

Usage Notes

- To run this procedure you must be logged in as the ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.
- To measure the impacts of a system change on a workload, you must ensure that the capture and replay systems are in the same logical state.
- Before initiating a workload capture, you should consider provisioning a refreshable clone to ensure the same start point for the replay.

> **Note:**
>
> You must subscribe to the Information event `com.oraclecloud.databaseservice.autonomous.database.information` to be notified at the start of `START_WORKLOAD_CAPTURE`. See Information Events on Autonomous Database for more information.

DBMS_CLOUD_ADMIN Exceptions

The following table describes exceptions for `DBMS_CLOUD_ADMIN`.

Exception	Code	Description
`invalid_service`	20001	An invalid service was specified.
`service_not_exist`	20002	A service specified does not exist.
`default_service`	20003	A service specified cannot be modified.
`invalid_char_set`	20029	Missing precondition or invalid (national) character set.
`invalid_enc_key_attr`	20030	Missing or invalid argument for key management.
`Already Using Oracle Managed Key`	000000	The Database is already using an Oracle managed key. You are trying to call the procedure while already using an Oracle managed key.
`Argument Provided for the procedure`	ORA-0000	An argument is provided for the procedure. Expected error message:No arguments required for this procedure.

DBMS_CLOUD_AI Package

The `DBMS_CLOUD_AI` package facilitates and configures the translation of natural language prompts to SQL statements.

- Summary of DBMS_CLOUD_AI Subprograms
  This section covers the `DBMS_CLOUD_AI` subprograms provided with Autonomous Database.

Summary of DBMS_CLOUD_AI Subprograms

This section covers the `DBMS_CLOUD_AI` subprograms provided with Autonomous Database.

Subprogram	Description
CREATE_PROFILE Procedure	This procedure new AI profile for translating natural language prompts to SQL statements.

Subprogram	Description
DISABLE_PROFILE Procedure	This procedure disables an AI profile in the current database.
DROP_PROFILE Procedure	This procedure drops an existing AI profile.
ENABLE_PROFILE Procedure	This procedure enables an AI profile to use in the current database.
GENERATE Function	This function generates a SQL statement using AI to translate.
GET_PROFILE Function	This function returns the profile name used in the current session.
GET_PROFILE Procedure	This procedure returns the profile name and the owner of the profile in the current session.
SET_ATTRIBUTE Procedure	This procedure sets AI profile attributes.
SET_PROFILE Procedure	This procedure sets AI profile for the current database.

- CREATE_PROFILE Procedure
  The procedure creates a new AI profile for translating natural language prompts to SQL statement.
- DROP_PROFILE Procedure
  The procedure drops an existing AI profile. If the profile does not exist, then the procedure throws an error.
- ENABLE_PROFILE Procedure
  This procedure enables the AI profile that the user specifies. The procedure changes the status of the AI profile to `ENABLED`.
- DISABLE_PROFILE Procedure
  This procedure disables the AI profile in the current database. The status of the AI profile is changed to `DISABLED` by this procedure.
- GET_PROFILE Procedure
  This procedure returns the AI profile name set in the current session.
- GET_PROFILE Procedure
  This procedure returns the AI profile name and the owner set in the current session.
- SET_ATTRIBUTE Procedure
  This procedure enables you to set AI profile attributes.
- SET_PROFILE Procedure
  This procedure sets AI profile for current session.
- GENERATE Function
- Profile Attributes

CREATE_PROFILE Procedure

The procedure creates a new AI profile for translating natural language prompts to SQL statement.

Syntax

```
DBMS_CLOUD_AI.CREATE_PROFILE
   profile_name        IN  VARCHAR2,
```

```
   attributes          IN  CLOB      DEFAULT NULL,
   status              IN  VARCHAR2  DEFAULT NULL,
   description         IN  CLOB      DEFAULT NULL
);
```

Parameters

Parameter	Description
profile_name	A name for the AI profile. The profile name must follow the naming rules of Oracle SQL identifier. Maximum length of profile name is 125 characters. This is a mandatory parameter.
attributes	Profile attributes in JSON format. See AI Profile Attributes for more details. The default value is NULL.
status	Status of the profile. The default value is enable.
description	Description for the AI profile. The default value is NULL.

Example

```
BEGIN
     DBMS_CLOUD_AI.CREATE_PROFILE(
          profile_name    => 'OpenAI',
          attributes      => JSON_OBJECT('provider' value 'openai',
                                         'credential_name' value
'openai_cred'),
          description     => 'AI profile to use OpenAI for SQL translation'
     );
END;
/
```

DROP_PROFILE Procedure

The procedure drops an existing AI profile. If the profile does not exist, then the procedure throws an error.

Syntax

```
DBMS_CLOUD_AI.DROP_PROFILE(
       profile_name        IN   VARCHAR2,
       force               IN   BOOLEAN DEFAULT FALSE
 );
```

Parameters

Parameter	Description
profile_name	Name of the AI profile

Parameter	Description
force	If TRUE, then the procedure ignores errors if AI profile does not exist. The default value for this parameter is FALSE.

Example

```
BEGIN
     DBMS_CLOUD_AI.DROP_PROFILE(profile_name => 'OPENAI');
END;
/
```

Usage Notes

Use force to drop a profile and ignore errors if AI profile does not exist.

ENABLE_PROFILE Procedure

This procedure enables the AI profile that the user specifies. The procedure changes the status of the AI profile to ENABLED.

Syntax

```
DBMS_CLOUD_AI.ENABLE_PROFILE(
     profile_name          IN   VARCHAR2
 );
```

Parameters

Parameter	Description
profile_name	Name for the AI profile to enable This parameter is mandatory.

Example to Enable AI Profile

```
BEGIN
     DBMS_CLOUD_AI.ENABLE_PROFILE(
         profile_name    => 'OPENAI'
     );
END;
/
```

DISABLE_PROFILE Procedure

This procedure disables the AI profile in the current database. The status of the AI profile is changed to DISABLED by this procedure.

Syntax

```
DBMS_CLOUD_AI.DISABLE_PROFILE(
      profile_name   IN  VARCHAR2
);
```

Parameters

Parameter	Description
profile_name	Name for the AI profile. This parameter is mandatory.

Example

```
BEGIN
     DBMS_CLOUD_AI.DISABLE_PROFILE(
         profile_name    => 'OPENAI'
     );
END;
/
```

GET_PROFILE Procedure

This procedure returns the AI profile name set in the current session.

Syntax

```
DBMS_CLOUD_AI.GET_PROFILE(
    profile_name      IN  VARCHAR2,
);
```

Parameters

Parameter	Description
profile_name	A name for the AI profile in the current session. This parameter is mandatory.

Example

```
   SELECT
        DBMS_CLOUD_AI.GET_PROFILE
        from DUAL;
```

GET_PROFILE Procedure

This procedure returns the AI profile name and the owner set in the current session.

Syntax

```
DBMS_CLOUD_AI.GET_PROFILE(

      profile_name  OUT  VARCHAR2,
      profile_owner OUT  VARCHAR2
);
```

Parameters

Parameter	Description
profile_name	A name for the AI profile in the current session. This parameter is mandatory.
profile_owner	Identifies the owner of the AI profile in the current session.

Example

This example shows how you can display the name and owner of the profile in the current session.

```
DECLARE
       l_profile_name  DBMS_ID;
       l_profile_owner DBMS_ID;
     BEGIN
       DBMS_CLOUD_AI.GET_PROFILE(profile_name => l_profile_name,
                                 profile_owner => l_profile_owner);
     END;
```

SET_ATTRIBUTE Procedure

This procedure enables you to set AI profile attributes.

Syntax

```
DBMS_CLOUD_AI.SET_ATTRIBUTE(
      profile_name         IN   VARCHAR2,
      attribute_name       IN   VARCHAR2,
      attribute_value      IN   CLOB
);
```

Parameters

Only the owner can set or modify the attributes of the AI profile. For a list of supported attributes, see Profile Attributes.

Parameter	Description
profile_name	Name of the AI profile for which you want to set the attributes. This parameter is mandatory.
attribute_name	Name of the AI profile attribute This parameter is mandatory.

Parameter	Description
attribute_value	Value of the profile attribute. The default value is NULL.

Example

```
BEGIN
 DBMS_CLOUD_AI.SET_ATTRIBUTE(
   profile_name    => 'OPENAI',
   attribute_name  => 'credential_name',
   attribute_value => 'OPENAI_CRED_NEW'
 );
END;
/
```

SET_PROFILE Procedure

This procedure sets AI profile for current session.

After setting an AI profile for the database session, any SQL statement with the prefix `SELECT AI` is considered a natural language prompt. Depending on the action specified with the `AI` prefix, a response is generated using AI. To use the `AI` prefix, see Examples of Using Select AI. Optionally, it is possible to override the profile attributes or modify attributes by specifying them in JSON format. See SET_ATTRIBUTE Procedure for setting the attributes.

The AI profile can only be set for current session if the owner of the AI profile is the session user.

To set an AI profile for all sessions of a specific database user or all user sessions in the database, consider using a database event trigger for `AFTER LOGON` event on the specific user or the entire database. See CREATE TRIGGER Statement for more details.

Syntax

```
DBMS_CLOUD_AI.SET_PROFILE(
    profile_name      IN  VARCHAR2,
);
```

Parameters

Parameter	Description
profile_name	A name for the AI profile in the current session. This parameter is mandatory.

Example

```
BEGIN
     DBMS_CLOUD_AI.SET_PROFILE(
       profile_name    => 'OPENAI'
     );
```

```
END;
/
```

GENERATE Function

This function provides AI translation in a stateless manner. With your existing AI profile, you can use this function to perform the supported actions such as `showsql`, `narrate`, or `chat`. The default action is `showsql`.
Overriding some or all of the profile attributes is also possible using this function.

Syntax

```
DBMS_CLOUD_AI.GENERATE(
    prompt          IN  CLOB,
    profile_name    IN  VARCHAR2 DEFAULT NULL,
    action          IN  VARCHAR2 DEFAULT NULL,
    attributes      IN  CLOB     DEFAULT NULL
) RETURN CLOB;
```

Parameters

Parameter	Description
`prompt`	Natural language prompt to translate using AI. The prompt can include `SELECT AI <action>` as the prefix. The action can also be supplied separately as an "action" parameter. The *action* supplied in prompt overrides the "action" parameter. Default action is `showsql`. This parameter is mandatory.

Parameter	Description
profile_name	Name of the AI profile. This parameter is optional if an AI profile is already set in the session using DBMS_CLOUD_AI.SET_PROFILE. The default value is NULL. The following conditions apply: • If a profile is set in the current session, the user may omit profile_name argument in the DBMS_CLOUD_AI.GENERATE function. • If the profile_name argument is supplied in the DBMS_CLOUD_AI.GENERATE function, it overrides any value set in the session using the DBMS_CLOUD_AI.SET_PROFILE procedure. • If there is no profile set in the session using the DBMS_CLOUD_AI.SET_PROFILE procedure, the profile_name argument must be supplied in the DBMS_CLOUD_AI.GENERATE function.

Note:

For Database Actions, you can either specify profile_name argument in DBMS_CLOUD_AI.GENERATE or you can run two steps as a PL/SQL script: DBMS_CLOUD_AI.SET_PROFILE and DBMS_CLOUD_AI.GENERATE.

```
EXEC
DBMS_CLOUD_AI.set_profile('OPENAI')
;

-----------------------------------
------------
SELECT
DBMS_CLOUD_AI.GENERATE(prompt
 => 'how many customers',

profile_name => 'OPENAI',

action       => 'showsql')
FROM dual;

-----------------------------------
-------------
SELECT
DBMS_CLOUD_AI.GENERATE(prompt
 => 'how many customers',

profile_name => 'OPENAI',

action       => 'narrate')
FROM dual;
-----------------------------------
--------
SELECT
```

Parameter	Description
	`` DBMS_CLOUD_AI.GENERATE(prompt => 'what is oracle autonomous database', profile_name => 'OPENAI', action => 'chat') FROM dual; ``` See Executing SQL Statements in the Code Editor for more information.
`action`	Action for translating natural prompt using AI. The supported actions include `showsql` (default), `narrate`, and `chat`. Descriptions of actions are included in Use AI Keyword to Enter Prompts. **Note:** This function does not support the `runsql` action. If you supply the `runsql` action, it returns the following error: ``` ORA-20000: runsql action is not supported by generate function ORA-06512: at "C##CLOUD$SERVICE.DBMS_CLOUD", line xxxx ORA-06512: at "C##CLOUD$SERVICE.DBMS_CLOUD_AI", line 2696 ORA-06512: at line x ```
`attributes`	Override specific AI profile attributes by supplying attributes in JSON format. See Profile Attributes for more details.

Examples

The following examples illustrate `showsql`, `narrate`, and `chat` actions that can be used with the `DBMS_CLOUD_AI.GENERATE` function.

An example with `showsql` action is as follows:

```
SELECT DBMS_CLOUD_AI.GENERATE(prompt       => 'how many customers',
                              profile_name => 'OPENAI',
                              action       => 'showsql')
FROM dual;
```

An example with `narrate` action is as follows:

```
SELECT DBMS_CLOUD_AI.GENERATE(prompt       => 'how many customers',
                              profile_name => 'OPENAI',
                              action       => 'narrate')
FROM dual;
```

An example with `chat` action is as follows:

```
SELECT DBMS_CLOUD_AI.GENERATE(prompt       => 'what is oracle autonomous
database',
                              profile_name => 'OPENAI',
                              action       => 'chat')
FROM dual;
```

Profile Attributes

Attributes of an AI profile help to manage and configure the behavior of the AI profile. Some attributes are optional and have a default value.

Attributes

Attribute Name	Description
`azure_deployment_name`	Name of the Azure OpenAI Service deployed model. The name can only include alphanumeric characters, underscore character (_) and a hyphen (-) character. The name cannot end with an underscore (_) or a hyphen (-). To know how to get the `azure_deployment_name`, see Create and deploy an Azure OpenAI Service resource.
`azure_resource_name`	Name of the Azure OpenAI Service resource. The resource name can only include alphanumeric characters and hyphens, and can't start or end with a hyphen. To know how to get the `azure_resource_name`, see Create and deploy an Azure OpenAI Service resource.
`comments`	Include column comments in the metadata used for translating natural language prompts using AI. `BOOLEAN` datatype is supported. The valid values are `TRUE` or `FALSE` for a string with `VARCHAR2` datatype. **Note:** Boolean values are not applicable in the `DBMS_CLOUD_AI.SET_ATTRIBUTE` procedure when setting a single attribute because `attribute_value` parameter is of `CLOB` datatype.
`conversation`	A `VARCHAR2` attribute that indicates if conversation history is enabled for a profile. Only OpenAI and Azure OpenAI Service support conversation history. Valid values are `true` or `false`. The default value is `false`. The values are not case sensitive.

Attribute Name	Description
credential_name	The name of the credential to access the AI provider APIs. Credential using bearer tokens can be created by using the provider name as the user name and bearer token as the password. Vault Secret credentials are also supported. Principle authentication, for example, Azure service principle, is also supported. For more information on how to configure it, see Use Azure Service Principal to Access Azure Resources. This is a mandatory attribute. See CREATE_CREDENTIAL Procedure.
max_tokens	Denotes the number of tokens to predict per generation. Default is 1024. See BPE Tokens for more details.
model	The name of the AI model being used to generate translation. Supported models: • For OpenAI: gpt-4, gpt-4-0613, gpt-4-32k, gpt-4-32k-0613, gpt-3.5-turbo (default), gpt-3.5-turbo-0613, gpt-3.5-turbo-16k, gpt-3.5-turbo-16k-0613 • For Cohere: command (default), command-nightly (experimental), command-light, and command-light-nightly (experimental). Smaller, "light" models are faster, while larger models will perform better. Custom models can also be supplied with their full ID. • For OCI Generative AI: cohere.command (default). Pretrained models for OCI Generative AI are all supported by Select AI. Custom models can also be supplied with their full OCIDs. To know more about supported models in OCI Generative AI, see Pretrained Foundational Models in Generative AI. **Note:** This parameter is not used for Azure as the model is determined when you create your deployment in the Azure OpenAI Service portal.

Attribute Name	Description
object_list	Array of JSON objects specifying the owner and object names that are eligible for natural language translation to SQL. To include all objects of a given user, omit the "name" and only specify the "owner" key in the JSON object. For translation natural language to SQL, the object name, object owner, object columns and comments are sent to the AI provider using HTTPS requests. Avoid specifying objects with sensitive object name, column names or comments in the object list. AI providers may have limit on the size of metadata allowed in translation requests. Consider limiting the list of objects suitable for the natural language prompts by your application users. **Format:** `[` `  {"owner": "SH", "name": "SALES",` `  {"owner": "TEST_USER"}` `]` External tables created using sync of OCI Data Catalog or AWS Glue can also be used the object list. This helps in managing metadata in central Data Catalogs and use the metadata directly for translating natural language prompts using AI.
oci_compartment_id	Specifies the OCID of the compartment you are permitted to access when calling the OCI Generative AI service. The compartment ID can contain alphanumeric characters, hyphens and dots. The default is the compartment ID of the PDB.
oci_endpoint_id	This attributes indicates the endpoint OCID of the Oracle dedicated AI hosting cluster. The endpoint ID can contain alphanumeric characters, hyphens and dots. To find the endpoint OCID, see Getting an Endpoint's Details in Generative AI. When you want to use the Oracle dedicated AI cluster, you must provide the endpoint OCID of the hosting cluster. By default, the endpoint ID is empty and the model is on-demand on a shared infrastructure.
oci_runtimetype	This attribute indicates the runtime type of the provided model. This attribute is required when the `model` attribute is specified. All allowed values can be found in OCI Generative AI runtimeType. See LlmInferenceRequest Reference. The default value is `COHERE` as the default model for OCI Generative AI is `cohere.command`. This option is applicable for all actions of Select AI. Select `LLAMA` if you are using the `meta.llama-2-70b-chat` model. Note that this model is suitable for the `chat` action of Select AI only.
provider	AI provider for the AI profile. Supported providers: • openai • cohere • azure • oci This is a mandatory attribute.

Attribute Name	Description
region	This attribute indicates the location of the Generative AI cluster that you want to use. The region can contain alphanumeric characters and hyphen characters. **Note:** The Oracle Generative AI cluster is only available in the Chicago region. The default region is us-chicago-1.
stop_tokens	The generated text will be terminated at the beginning of the earliest stop sequence. Sequence will be incorporated into the text. The attribute value must be a valid array of string values in JSON format. stop_tokens takes a JSON array as input. To learn more about stop tokens or stop sequences, see OpenAI or Cohere documentation.
temperature	Sampling from generation models incorporates randomness, so that the same prompt may yield different outputs each time you hit "generate". Temperature is a non-negative float number used to tune the degree of randomness. Lower temperatures mean less random generations. See Temperature for more details. This parameter is applicable to all the supported service providers.

The following example is using Cohere as the provider and displays custom profile attributes:

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
     profile_name => 'COHERE',
     attributes =>
      '{"provider": "cohere",
        "credential_name": "COHERE_CRED",
        "object_list": [{"owner": "ADMIN"}],
        "max_tokens":512,
        "stop_tokens": [";"],
        "model": "command-nightly",
        "temperature": 0.5,
        "comments": true
       }');
END;
/
```

The following example shows custom profile attributes using OCI Generative AI:

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
      'GENAI',
      '{"provider":
"oci",
        "credential_name": "GENAI_CRED",
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "countries"},
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
```

```
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}],
          "oci_compartment_id": "ocid1.compartment.oc1...",
      "oci_endpoint_id": "ocid1.generativeaiendpoint.oc1.us-chicago-1....",
      "region": "us-chicago-1",
      "model": "cohere.command-light",
      "oci_runtimetype": "COHERE"
         }');
END;
/
```

DBMS_CLOUD_FUNCTION Package

The `DBMS_CLOUD_FUNCTION` package allows you invoke OCI and AWS Lambda remote functions in your Autonomous Database as SQL functions.

- Summary of DBMS_CLOUD_FUNCTION Subprograms
  This table summarizes the subprograms included in the `DBMS_CLOUD_FUNCTION` package.

Summary of DBMS_CLOUD_FUNCTION Subprograms

This table summarizes the subprograms included in the `DBMS_CLOUD_FUNCTION` package.

Subprogram	Description
CREATE_CATALOG Procedure	This procedure creates a catalog.
CREATE_FUNCTION Procedure	This procedure creates functions in a catalog.
DROP_CATALOG Procedure	This procedure drops a catalog and functions created using the catalog
DROP_FUNCTION Procedure	This procedure drops functions from a catalog.
LIST_FUNCTIONS Procedure	This procedure lists all the functions in a catalog.
SYNC_FUNCTIONS Procedure	This procedure creates a PL/SQL wrapper for adding new functions to the catalog and removing wrappers for functions that have been deleted from the catalog.

- CREATE_CATALOG Procedure
  This procedure creates a catalog in the database. The `DBMS_CLOUD_FUNCTION.CREATE_CATALOG` procedure creates a catalog. A catalog is a set of functions that creates the required infrastructure to execute subroutines. This procedure is overloaded.
- CREATE_FUNCTION Procedure
  This procedure creates functions in a catalog. There are two overloaded `DBMS_CLOUD_FUNCTION.CREATE_FUNCTION` procedures.
- DROP_CATALOG Procedure
  The `DBMS_CLOUD_FUNCTION.DROP_CATALOG` procedure drops the catalog and functions created using the catalog. This procedure is overloaded.
- DROP_FUNCTION Procedure
  The `DBMS_CLOUD_FUNCTION.DROP_FUNCTION` procedure drops the function. This procedure is overloaded.
- LIST_FUNCTIONS Procedure
  This procedure lists all the functions in a catalog.

- SYNC_FUNCTIONS Procedure
  This procedure creates a PL/SQL wrapper for adding new functions to the catalog and removing wrappers for functions that have been deleted from the catalog.

CREATE_CATALOG Procedure

This procedure creates a catalog in the database. The `DBMS_CLOUD_FUNCTION.CREATE_CATALOG` procedure creates a catalog. A catalog is a set of functions that creates the required infrastructure to execute subroutines. This procedure is overloaded.

Syntax

```
DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
    credential_name             IN VARCHAR2,
    catalog_name                IN VARCHAR2,
    service_provider            IN VARCHAR2,
    cloud_params                IN CLOB
);

DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
    library_name                IN VARCHAR2,
    library_listener_url        IN VARCHAR2,
    library_wallet_dir_name     IN VARCHAR2,
    library_ssl_server_cert_dn  IN VARCHAR2,
    library_remote_path         IN VARCHAR2
);
```

Parameters

Parameter	Description
`credential_name`	Specifies the name of the credential for authentication. This parameter is mandatory.
`service_provider`	Specifies the type of the service provider. This parameter can have `OCI` or `AWS` or `AZURE` as a parameter value. This parameter is mandatory.
`catalog_name`	Specifies the catalog name. This parameter is mandatory.
`cloud_params`	Provides parameter to the function. For example, Compartment OCID, Regions and Azure subscription_id. This parameter is mandatory.
`library_name`	Specifies the name of the library when creating a remote library. This parameter is mandatory.
`library_listener_url`	Specifies the remote location of the library. The parameter accepts a String value in *`host_name:port_number`* format. For example: `EHRPMZ_DBDOMAIN.adb-us-phoenix1.com:16000` This parameter is mandatory.

Parameter	Description
library_remote_path	Specifies the remote library path. You must provide the full absolute path to the remote library. For example:/u01/app/oracle/product/21.0.0.0/client_1/lib/libst_shape.so This parameter is mandatory.
library_wallet_dir_name	Specifies the directory where the self-signed wallet is stored. This parameter is mandatory.
library_ssl_server_cert_dn	Specifies the server certificate Distinguished Name (DN). This parameter is mandatory.

Errors

Error Code	Description
ORA-20000	This error is raised in either of the following conditions: • cloud_params value is missing or incorrect parameter values are passed. • library_name value is not unique when creating a library.
ORA-20001	This error is raised in either of the following conditions: • The credential referenced in the credential_name does not exist. • The Listener specified at the library_listener_url is not reachable when creating a library.
ORA-20002	This error is raised in either of the following conditions: • This error is raised when the catalog already exists. • The specified Server certificate directory is empty when creating a library.
ORA-20009	This error is raised when the service provider doesn't exist.

Examples

```
BEGIN
    DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
        credential_name  => 'DEFAULT_CREDENTIAL',
        catalog_name     => 'OCI_DEMO_CATALOG',
        service_provider => 'OCI',
        cloud_params     => ("region_id":"us-phoenix-1",
"compartment_id":"compartment_id"
 );
END;
/


BEGIN
    DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
        credential_name  => 'AZURE$PA',
        catalog_name     => 'AZURE_DEMO_CATALOG',
        service_provider => 'AZURE',
        cloud_params     => '{"subscription_id":"44495e6a-8ff1-4161-
b387-0e14e675b878"}'
 );
```

```
END;
/


BEGIN
    DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
    library_name               => 'EXT_DEMOLIB',
    library_listener_url       => 'remote_extproc_hostname:16000',
    library_wallet_dir_name    => 'WALLET_DIR',
    library_ssl_server_cert_dn => 'CN=VM Hostname',
    library_remote_path        => '/u01/app/oracle/extproc_libs/library name'
 );
END;
/
```

Usage Note

- To create a catalog you must be logged in as the `ADMIN` user or have privileges on the following:
  - `DBMS_CLOUD_OCI_FNC_FUNCTIONS_INVOKE`
  - `DBMS_CLOUD_OCI_FNC_FUNCTIONS_INVOKE_INVOKE_FUNCTION_RESPONSE_T`
  - `DBMS_CLOUD`
  - Read privilege on `USER_CLOUD_FUNCTION`
  - Read privilege on `USER_CLOUD_FUNCTION_CATALOG`

CREATE_FUNCTION Procedure

This procedure creates functions in a catalog. There are two overloaded `DBMS_CLOUD_FUNCTION.CREATE_FUNCTION` procedures.

The `DBMS_CLOUD_FUNCTION.CREATE_FUNCTION` procedure is only supported for cloud functions.

`CREATE_FUNCTION` Syntax

```
DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
    credential_name   IN VARCHAR2,
    catalog_name      IN VARCHAR2,
    function_name     IN VARCHAR2,
    function_id       IN VARCHAR2,
    input_args        IN CLOB      DEFAULT NULL,
    return_type       IN VARCHAR2  DEFAULT 'CLOB',
    response_handler  IN VARCHAR2  DEFAULT NULL
);
```

Response Handler signature

```
<USER DEFINED TYPE> response_handler_name(function_response in CLOB)RETURNS
CLOB;
```

The return type of this is user defined type or PL/SQL type. The `function_response` is of JSON with fields.

```
'{
"STATUS":"<RESPONCE STATUS>",
"RESPONSE_BODY":"<FUNCTION RESPONSE>"
}'
```

`CREATE_FUNCTION` Parameters

Parameter	Description
`credential_name`	Specifies the name of the credential for authentication. This parameter is mandatory.
`catalog_name`	Specifies the catalog name. This parameter is mandatory.
`function_name`	Specifies the PL/SQL function name. This parameter is mandatory.
`function_id`	The `function_id` parameter value refers to the OCI function, AWS Lambda or Azure function. This parameter is mandatory.
`input_args`	Specifies the key value JSON pair accepting input arguments and their types.
`return_type`	Defines the return type of the function. The return type is of `CLOB` data type.
`response_handler`	Specifies the user defined callback to handle response.

`CREATE_FUNCTION` Exceptions

Error Code	Description
`ORA-20001`	This error is raised when the credential referenced in the `credential_name` does not exist.
`ORA-20003`	This error is raised when the specified catalog does not exist.
`ORA-20004`	This error is raised when the specified function already exists.
`ORA-20005`	This error is raised when the function ID or function Amazon Resource Names (ARN) does not exist.
`ORA-20006`	This error is raised when the input arguments are invalid.
`ORA-20007`	This error is raised when the return type is missing or invalid.
`ORA-20008`	This error is raised when the response handler is missing or invalid.

`CREATE_FUNCTION` Example

```
VAR function_args CLOB;
EXEC :function_args := TO_CLOB('{"command": "VARCHAR2", "value":
"VARCHAR2"}');
BEGIN
    DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
    credential_name  => 'DEFAULT_CREDENTIAL',
       catalog_name     => 'OCI_DEMO_CATALOG',
```

```
       function_name    => 'demo_function',
       function_id      =>
'ocid1.fnfunc.oc1.phx.aaaaaaaazkrbjv6ntowwxlbbp5ct4otsj4o2hdw4ayosyosv4sthmya2
lyza',
       input_args       => :function_args);
 );
END;
/
```

`CREATE_FUNCTION` Syntax

```
DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
  library_name          IN VARCHAR2,
  function_name         IN VARCHAR2,
  function_id           IN VARCHAR2 DEFAULT NULL,
  plsql_params          IN CLOB DEFAULT NULL,
  external_params       IN CLOB DEFAULT NULL,
  api_type              IN VARCHAR2 DEFAULT 'FUNCTION',
  with_context          IN BOOLEAN DEFAULT FALSE,
  return_type           IN VARCHAR2 DEFAULT NULL
);
```

`CREATE_FUNCTION` Parameters

Parameter	Description
`library_name`	Specifies the remote library name. This parameter is mandatory.
`function_name`	Specifies the PL/SQL function name. This parameter is mandatory.
`function_id`	The `function_id` parameter value refers to the external procedures (`extproc`). If the value for `function_id` is not provided, the value in the `function_name` is used.
`plsql_params`	Specifies the key value JSON pair accepting the parameters for the PL/SQL wrapper. The values must be provided in the `"var_name":"modetype, datatype"` format. • `var_name` : is the name of the variable. This parameter is mandatory. • `modetype` : specifies the variable mode. The variable mode can be one of the following: – `IN` – `OUT` – `IN OUT` • `datatype` : specifies the variable datatype. This parameter is mandatory. The default value for `plsql_params` is `NULL`.
`external_params`	Specifies the parameters that need to be provided to the external C function. If value is not provided for `external_params`, the PL/SQL parameters are used.
`api_type`	Specifies the type of API (function or procedure). The default value for `api_type` is `function`.

Parameter	Description
with_context	Specifies that a context pointer is passed to the external procedure. This context is used by the external C library for connecting back to the database. The default value for with_context is FALSE.
return_type	Specifies the return type of the function created.

CREATE_FUNCTION Example

```
DECLARE
    plsql_params clob    := TO_CLOB('{"sal": "IN, FLOAT", "comm" :"IN,
FLOAT"}');
    external_params clob := TO_CLOB('sal FLOAT, sal INDICATOR SHORT, comm
FLOAT, comm INDICATOR SHORT,
    RETURN INDICATOR SHORT, RETURN FLOAT');
BEGIN
DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
    LIBRARY_NAME    => 'demolib',
    FUNCTION_NAME   => '"PercentComm"',
    PLSQL_PARAMS    => plsql_params,
    EXTERNAL_PARAMS => external_params,
    API_TYPE        => 'FUNCTION',
    WITH_CONTEXT    => FALSE,
    RETURN_TYPE     => 'FLOAT'
);
END;
/
```

DROP_CATALOG Procedure

The DBMS_CLOUD_FUNCTION.DROP_CATALOG procedure drops the catalog and functions created using the catalog. This procedure is overloaded.

Syntax

```
DBMS_CLOUD_FUNCTION.DROP_CATALOG (
    catalog_name      IN VARCHAR2
 );

DBMS_CLOUD_FUNCTION.DROP_CATALOG (
    library_name      IN VARCHAR2
 );
```

Parameters

Parameter	Description
catalog_name	Specifies the catalog name. This parameter is mandatory.
library_name	Specifies the library name. This parameter is mandatory.

Errors

Error Code	Description
ORA-20003	This error is raised when the specified catalog doesn't exist.

Example:

```
BEGIN
    DBMS_CLOUD_FUNCTION.DROP_CATALOG (
      catalog_name     => 'OCI_DEMO_CATALOG'
  );
END;
/
```

Example:

```
BEGIN
    DBMS_CLOUD_FUNCTION.DROP_CATALOG (
      library_name     => 'library_name'
  );
END;
/
```

DROP_FUNCTION Procedure

The DBMS_CLOUD_FUNCTION.DROP_FUNCTION procedure drops the function. This procedure is overloaded.

The DBMS_CLOUD_FUNCTION.DROP_FUNCTION procedure is only supported for cloud functions.

Syntax

```
DBMS_CLOUD_FUNCTION.DROP_FUNCTION (
    catalog_name      IN VARCHAR2,
    function_name     IN VARCHAR2
);

DBMS_CLOUD_FUNCTION.DROP_FUNCTION (
    library_name     IN VARCHAR2,
    function_name    IN VARCHAR2
);
```

Parameters

Parameter	Description
catalog_name	Specifies the catalog name. This parameter is mandatory.
function_name	Specifies the name of the function to be dropped. This parameter is mandatory.

Parameter	Description
library_name	Specifies the library name. This parameter is mandatory.

Examples

```
BEGIN
    DBMS_CLOUD_FUNCTION.DROP_FUNCTION (
       catalog_name     => 'OCI_DEMO_CATALOG',
       function_name    => 'demo_function');
END;
/


BEGIN
    DBMS_CLOUD_FUNCTION.DROP_FUNCTION (
       library_name     => 'EXTPROC_DEMO_LIBRARY',
       function_name    => 'demo_function');
END;
/
```

LIST_FUNCTIONS Procedure

This procedure lists all the functions in a catalog.

Syntax

```
DBMS_CLOUD_FUNCTION.LIST_FUNCTIONS (
    credential_name   IN VARCHAR2,
    catalog_name      IN VARCHAR2,
    function_list     OUT VARCHAR2
);
```

Parameters

Parameter	Description
credential_name	Specifies the name of the credential for authentication. This parameter is mandatory.
function_list	Returns the list of functions in JSON format. This parameter is mandatory.
catalog_name	Specifies the catalog name. This parameter is mandatory.

Errors

Error Code	Description
ORA-20000	This error is raised when cloud_params value is missing or incorrect parameter values are passed.

Error Code	Description
ORA-20001	This error is raised when the credential referenced in the credential_name does not exist.
ORA-20003	This error is raised when the specified catalog does not exist.

Example:

```
VAR function_list CLOB;
BEGIN
    DBMS_CLOUD_FUNCTION.LIST_FUNCTIONS (
    credential_name  => 'DEFAULT_CREDENTIAL',
    catalog_name     => 'OCI_DEMO_CATALOG',
    function_list    => :function_list);
 );
END;
/
SELECT JSON_QUERY(:function_list, '$' RETURNING VARCHAR2(32676) pretty) AS
search_results FROM dual;
```

SYNC_FUNCTIONS Procedure

This procedure creates a PL/SQL wrapper for adding new functions to the catalog and removing wrappers for functions that have been deleted from the catalog.

Syntax

```
DBMS_CLOUD_FUNCTION.SYNC_FUNCTIONS (
    catalog_name      IN VARCHAR2,
    refresh_rate     IN VARCHAR2 DEFAULT 'DAILY'
);
```

Parameters

Parameter	Description
catalog_name	Specifies the catalog name. This parameter is mandatory.
refresh_rate	Specifies the refresh rate of the function. refresh_rate can accept the following values: • HOURLY • DAILY • WEEKLY • MONTHLY The default value for this parameter is DAILY.

Errors

Error Code	Description
ORA-20003	This error is raised when the specified catalog does not exist.

Error Code	Description
ORA-20004	This error is raised when an invalid value is passed for the refresh_rate parameter.

Example:

```
BEGIN
    DBMS_CLOUD_FUNCTION.SYNC_FUNCTIONS (
       catalog_name     => 'OCI_DEMO_CATALOG'
 );
END;
/
```

DBMS_CLOUD_LINK Package

The DBMS_CLOUD_LINK package allows a user to register a table or a view as a data set for read only access with Cloud Links.

- DBMS_CLOUD_LINK Overview
  Describes the use of the DBMS_CLOUD_LINK package.
- Summary of DBMS_CLOUD_LINK Subprograms
  Shows a table with a summary of the subprograms included in the DBMS_CLOUD_LINK package.

DBMS_CLOUD_LINK Overview

Describes the use of the DBMS_CLOUD_LINK package.

The DBMS_CLOUD_LINK package provides the REGISTER procedure that allows you to register a table or a view as a data set for use with Cloud Links. Before you can register a data set, the ADMIN user must grant a user permission to register a data set using the DBMS_CLOUD_LINK_ADMIN.GRANT_REGISTER procedure. After the ADMIN runs GRANT_REGISTER, a user can register a table or a view they own as a registered data set (or register an object in another schema if the user has READ WITH GRANT OPTION privilege on the object). Registered data sets provide remote access to the registered object with Cloud Links, subject to the scope specified with the REGISTER procedure.

To run DBMS_CLOUD_LINK.REGISTER you have to have execute privilege on DBMS_CLOUD_LINK.REGISTER, in addition to having previously run DBMS_CLOUD_LINK_ADMIN.GRANT_REGISTER. Only the ADMIN user and schemas with PDB_DBA role have execute privilege on DBMS_CLOUD_LINK.REGISTER by default.

Summary of DBMS_CLOUD_LINK Subprograms

Shows a table with a summary of the subprograms included in the DBMS_CLOUD_LINK package.

Subprogram	Description
DESCRIBE Function	This function retrieves the description for a data set. The description is provided when a data set is registered with DBMS_CLOUD_LINK.REGISTER.

Subprogram	Description
FIND Procedure	Retrieves the namespace, name, and description for data sets that match the search string. Matching data sets are only shown if they are accessible to the user, based on access restrictions.
GET_DATABASE_ID Function	Returns a unique identifier for the Autonomous Database instance. Repeated calls to `DBMS_CLOUD_LINK.GET_DATABASE_ID` on the same instance always return the same value.
GRANT_AUTHORIZATION Procedure	Grants authorization to a specified database to access the specified data set.
REGISTER Procedure	Registers a table or view as data set.
REVOKE_AUTHORIZATION Procedure	Revokes authorization for a specified database to access the specified data set.
UNREGISTER Procedure	Removes a registered data set.

- DESCRIBE Function
- FIND Procedure
- GET_DATABASE_ID Function
- GRANT_AUTHORIZATION Procedure
- REGISTER Procedure
- REVOKE_AUTHORIZATION Procedure
- UNREGISTER Procedure

DESCRIBE Function

This function retrieves the description for a data set. The description is provided when a data set is registered with `DBMS_CLOUD_LINK.REGISTER`.

Syntax

```
DBMS_CLOUD_LINK.DESCRIBE(
      namespace         IN   VARCHAR2,
      name              IN   VARCHAR2
) return CLOB;
```

Parameters

Parameter	Description
`namespace`	Specifies the namespace of registered data set.
`name`	Specifies the name of a registered data set.

Usage Note

You can use this function subject to the access restrictions imposed at registration time with `DBMS_CLOUD_LINK.REGISTER`. If a data set is not accessible to a database, then its description will not be retrieved.

FIND Procedure

This procedure retrieves the namespace, name, and description for data sets that match the search string. Matching data sets are only shown if they are accessible to the user, based on access restrictions.

Syntax

```
DBMS_CLOUD_LINK.FIND(
      search_string         IN   VARCHAR2,
      search_result         OUT   CLOB
);
```

Parameters

Parameter	Description
`search_string`	Specifies the search string. The search string is not case sensitive.
`search_result`	A JSON document that includes the namespace, name, and description values for the data set.

Usage Note

The search string is not case sensitive and the package leverages free text search using Oracle Text.

GET_DATABASE_ID Function

The function returns a unique identifier for the Autonomous Database instance. Repeated calls to `DBMS_CLOUD_LINK.GET_DATABASE_ID` on the same instance always return the same value.

You can call this function on a database that is accessing a registered data set remotely to obtain the database ID. This allows you to provide the database iD so that a data set owner can leverage a more fine-grained data access control, for example with VPD, based on a specified database IDs from remote sites.

A database ID identifies each remote database that accesses a registered data set, to track and audit access in the V$CLOUD_LINK_ACCESS_STATS and GV$CLOUD_LINK_ACCESS_STATS Views on the database that owns a registered data set.

Syntax

```
DBMS_CLOUD_LINK.GET_DATABASE_ID()
  RETURN VARCHAR2;
```

Usage Notes

Cloud links use the unique identifier that `DBMS_CLOUD_LINK.GET_DATABASE_ID` returns to identify individual databases that are accessing a data set remotely. The database owning the registered data set tracks and audits the database ID as a record of the origin for data set access in the V$CLOUD_LINK_ACCESS_STATS and GV$CLOUD_LINK_ACCESS_STATS Views.

The `DBMS_CLOUD_LINK.GET_DATABASE_ID` identifier is available as a `SYS_CONTEXT` value so that you can programmatically obtain this information about a connecting remote session using

SYS_CONTEXT, to further restrict and control what specific data can be accessed remotely by individual Autonomous Database instances with Virtual Private Databases (VPD)s.

Return Values

A unique identifier for the Autonomous Database instance of VARCHAR2.

GRANT_AUTHORIZATION Procedure

This procedure grants authorization to a specified database to access the specified data set.

Syntax

```
DBMS_CLOUD_LINK.GRANT_AUTHORIZATION(
      database_id         IN   VARCHAR2,
      namespace           IN   VARCHAR2 DEFAULT,
      name                IN   VARCHAR2
);
```

Parameters

Parameter	Description
database_id	Specifies the database ID for an Autonomous Database instance. Use DBMS_CLOUD_LINK.GET_DATABASE_ID to obtain the database ID.
namespace	Specifies the data set namespace to grant access authorization for the specified database_id.
name	Specifies the data set name to grant access authorization for the specified database_id.

REGISTER Procedure

The procedure registers a table or a view as a data set to allow remote read only access, subject to restrictions imposed by the scope parameter.

Syntax

```
DBMS_CLOUD_LINK.REGISTER(
     schema_name      IN VARCHAR2,
     schema_object    IN VARCHAR2,
     namespace        IN VARCHAR2,
     name             IN VARCHAR2,
     description      IN CLOB,
     scope            IN CLOB,
     auth_required    IN BOOLEAN DEFAULT,
     data_set_owner   IN VARCHAR2 DEFAULT,
     offload_targets  IN CLOB DEFAULT

);
```

Parameters

Parameter	Description
schema_name	Specifies the owner of the table or view specified with the schema_object parameter.
schema_object	Specifies the name of a table or a view.
namespace	Specifies the namespace for the data set. A NULL value specifies a system-generated namespace value, unique to the Autonomous Database instance.
name	Specifies the name for the data set.
description	Specifies text to describe the data.
scope	Describes who is allowed to access the data set. The value is a comma separated list consisting of one or more of the following: • Database OCID: Access to the data set is allowed for the specific Autonomous Database instances identified by OCID. • Compartment OCID: Access to the data set is allowed for databases in the compartments identified by compartment OCID. • Tenancy OCID: Access to the data set is allowed for databases in the tenancies identified by tenancy OCID. • Region name: Access to the data set is allowed for databases in the region identified by the named region. By scope, Cloud Links access is limited to within a single region and is not cross-region. A consumer in a different region can access a data set only when a cross-region Refreshable Clone of the database that is the data set owner exists in the consumer database's region. See Register or Unregister a Data Set in a Different Region for more information. • MY$COMPARTMENT: Access to the data set is allowed for databases in the same compartment as the data set owner. • MY$TENANCY: Access to the data set is allowed for databases in the same tenancy as the data set owner. • MY$REGION: Access to the data set is allowed for databases in the same region as the data set owner. • List of regions: Access to the data set is allowed for databases in the specified regions For example: `scope => 'us-phoenix-1,us-ashburn-1',` A consumer in a different region can access a data set only when a cross-region Refreshable Clone of the database that is the data set owner exists in the consumer database's region. The scope values, MY$REGION, MY$TENANCY, and MY$COMPARTMENT are variables that act as convenience macros and resolve to OCIDs.

Parameter	Description
auth_required	Specifies that an additional authorization is required for databases to read from the data set. The following cases are possible: • Databases that are within the SCOPE specified and have been authorized with DBMS_CLOUD_LINK.GRANT_AUTHORIZATION can view rows from the data set. • Any databases that are within the specified SCOPE but have not been authorized with DBMS_CLOUD_LINK.GRANT_AUTHORIZATION cannot view data set rows. In this case, consumers without authorization see the data set as empty. • Databases that are not within the SCOPE specified see an error when attempting to access the data set.
data_set_owner	Specifies the data set owner. This is indicates who the data set belongs to or who is responsible for updating and maintaining the data set. For example, you can set the data_set_owner to the email address of the person who registered the data set.

Parameter	Description
offload_targets	Specifies one or more Autonomous Database OCIDs of refreshable clones where access to data sets is offloaded, from the Autonomous Database where the data set is registered. The offload_targets value is a JSON document that defines one or more CLOUD_LINK_DATABASE_ID and OFFLOAD_TARGET key value pairs: • CLOUD_LINK_DATABASE_ID is one of: – A database ID: This specifies a database ID for the data set consumer whose request is offloaded to the corresponding refreshable clone specified with the OFFLOAD_TARGET value. Obtain the database ID by running DBMS_CLOUD_LINK.GET_DATABASE_ID. See GET_DATABASE_ID Function for more information. – ANY: This specifies that any data set consumer's request is offloaded to the corresponding offload target. A consumer's data set request will be routed to the corresponding offload target. If you specify ANY without specifying database IDs, all data set requests from consumers are offloaded to the refreshable clone specified with the OFFLOAD_TARGET value. If you specify both database IDs and ANY, data set requests from consumers that do not match a database ID are offloaded to the refreshable clone specified with the OFFLOAD_TARGET value. • The OFFLOAD_TARGET is the OCID for an Autonomous Database instance that is a refreshable clone. For example, the following shows a JSON sample with three OFFLOAD_TARGET/CLOUD_LINK_DATABASE_ID value pairs: `{ "OFFLOAD_TARGETS": [{ "CLOUD_LINK_DATABASE_ID": "34xxxxx69708978", "OFFLOAD_TARGET": "ocid1.autonomousdatabase.oc1..xxxxx3pv6wkcr4jqae5f44n2b2m2yt2j6rx32uzr4h25vqstifsfabc" }, { "CLOUD_LINK_DATABASE_ID": "34xxxxx89898978", "OFFLOAD_TARGET": "ocid1.autonomousdatabase.oc1..xxxxx3pv6wkcr4jqae5f44n2b2m2yt2j6rx32uzr4h25vqstifsfdef" }, { "CLOUD_LINK_DATABASE_ID": "34xxxxx4755680", "OFFLOAD_TARGET": "ocid1.autonomousdatabase.oc1..xxxxx3pv6wkcr4jqae5f44n2b2m2yt2j6rx32uzr4h25vqstifsfghi" }] }` When a data set consumer requests access to a data set that you register with offload_targets with the ANY keyword, then any request for access is offloaded to the refreshable clone identified with OFFLOAD_TARGET in the

Parameter	Description
	supplied JSON (except requests that have an explicit entry in the supplied JSON). For example, the following shows a JSON sample with one explicit `OFFLOAD_TARGET`/`CLOUD_LINK_DATABASE_ID` value pair, and one `ANY` entry: `{` `  "OFFLOAD_TARGETS": [` `    {` `      "CLOUD_LINK_DATABASE_ID": "ANY",` `      "OFFLOAD_TARGET":` `"ocid1.autonomousdatabase.oc1..xxxxx3pv6wkcr4jqae5f44n2b2` `m2yt2j6rx32uzr4h25vqstifsfdef"` `    },` `    {` `      "CLOUD_LINK_DATABASE_ID": "34xxxxx4755680",` `      "OFFLOAD_TARGET":` `"ocid1.autonomousdatabase.oc1..xxxxx3pv6wkcr4jqae5f44n2b2` `m2yt2j6rx32uzr4h25vqstifsfghi"` `    }` `  ]` `}` `DBMS_CLOUD_LINK.REGISTER` reports an error if the OCID supplied as an `OFFLOAD_TARGET` value is not an OCID of a refreshable clone in the same region. See Use Refreshable Clones with Autonomous Database for information on using refreshable clones.

Usage Notes

- You can register an object in another schema if you have `READ WITH GRANT OPTION` privilege on the object.
- After you register an object, users may need to wait up to ten (10) minutes to access the object with Cloud Links.
- To change the scope for an existing data set, you must first unregister the data set with `DBMS_CLOUD_LINK.UNREGISTER`. Then you need to register the data set with the new scope with `DBMS_CLOUD_LINK.REGISTER`. The total wait time for these two operations can be up to 20 minutes, including 10 minutes for the unregister and another 10 minutes for the register to be propagated and accessible through Cloud Links.

  This delay can also impact the visibility of the data set in the both the `DBA_CLOUD_LINK_REGISTRATIONS` and `DBA_CLOUD_LINK_ACCESS` views.
- You can register a table or view that resides in another user's schema when the you have `READ WITH GRANT OPTION` privileges for the table or view.
- The scope you set when you register a data set is only honored when it matches or is more restrictive than the value set with `DBMS_CLOUD_LINK_ADMIN.GRANT_REGISTER`. For example, assume the ADMIN granted the scope `'MY$TENANCY'` with `GRANT_REGISTER`, and

the user specifies `'MY$REGION'` when they register a data set with `DBMS_CLOUD_LINK.REGISTER`. In this case they would see an error such as the following:

```
ORA-20001: Share privileges are not enabled for current user or it is
enabled but not for scope MY$REGION
```

- Certain hierarchical validity checks for registration cannot happen at registration time. Invalid registrations will not be visible, discoverable, or even accessible to anyone.
- To use `DBMS_CLOUD_LINK.REGISTER`, you must have execute privilege on the `DBMS_CLOUD_LINK` package, in addition to the register privilege assigned with `DBMS_CLOUD_LINK_ADMIN.GRANT_REGISTER`. Only the ADMIN user and schemas with `PDB_DBA` have this privilege by default.
- When you register a data set on a refreshable clone in a remote region, the invocation of `DBMS_CLOUD_LINK.REGISTER` on the remote region clone must use the same parameters with the same values as on the source database, with the exception of the `offload_targets` parameter.

  For example, when you run `DBMS_CLOUD_LINK.REGISTER` with scope set to `MY$COMPARTMENT` on the source Autonomous Database instance, run the procedure again on the cross-region refreshable clone with the same scope parameter value (`MY$COMPARTMENT`).
- If you specify the `offload_targets` parameter with `DBMS_CLOUD_LINK.REGISTER` on source, you should omit this parameter when you register the data set on a cross-region refreshable clone.

REVOKE_AUTHORIZATION Procedure

This procedure revokes authorization for a specified database to access the specified data set.

Syntax

```
DBMS_CLOUD_LINK.REVOKE_AUTHORIZATION(
      database_id         IN   VARCHAR2,
      namespace           IN   VARCHAR2 DEFAULT,
      name                IN   VARCHAR2
);
```

Parameters

Parameter	Description
`database_id`	Specifies the database ID for an Autonomous Database instance. Use `DBMS_CLOUD_LINK.GET_DATABASE_ID` to obtain the database ID.
`namespace`	Specifies the data set namespace to revoke access authorization for the specified `database_id`.
`name`	Specifies the data set name to revoke access authorization for the specified `database_id`.

UNREGISTER Procedure

The procedure allows a user who previously registered a table or a view with the `REGISTER` procedure to unregister the table or view so that it no longer is available for remote access.

Syntax

```
DBMS_CLOUD_LINK.UNREGISTER(
      namespace         IN   VARCHAR2,
      name              IN   VARCHAR2
);
```

Parameters

Parameter	Description
`namespace`	Specifies a username.
`name`	Specifies the name for the data set.

Usage Note

`DBMS_CLOUD_LINK.UNREGISTER` may also take up to ten (10) minutes to fully propagate, after which the data can longer be accessed remotely.

DBMS_CLOUD_LINK_ADMIN Package

The `DBMS_CLOUD_LINK_ADMIN` package allows the ADMIN user to enable a database user to register data sets or to access registered data sets for a given Autonomous Database instance, subject to the access restrictions as defined with the granted scope.

Privileges can also be disabled for a user that has the privileges set to register data sets or access registered data sets.

- DBMS_CLOUD_LINK_ADMIN Overview
  Describes use of the `DBMS_CLOUD_LINK_ADMIN` package.
- Summary of DBMS_CLOUD_LINK_ADMIN Subprograms
  This table summarizes the subprograms included in the `DBMS_CLOUD_LINK_ADMIN` package.

DBMS_CLOUD_LINK_ADMIN Overview

Describes use of the `DBMS_CLOUD_LINK_ADMIN` package.

Cloud Links provide a cloud-based method to remotely access read only data on an Autonomous Database instance. The `DBMS_CLOUD_LINK_ADMIN` package leverages Oracle Cloud Infrastructure access mechanisms to make data sets accessible within a specific scope and in addition there is a optional authorization step.

Summary of DBMS_CLOUD_LINK_ADMIN Subprograms

This table summarizes the subprograms included in the `DBMS_CLOUD_LINK_ADMIN` package.

Subprogram	Description
GRANT_AUTHORIZE Procedure	Allows the ADMIN user to revoke a user's permission to invoke `DBMS_CLOUD_LINK.GRANT_AUTHORIZATION` and `DBMS_CLOUD_LINK.REVOKE_AUTHORIZATION` procedures.
GRANT_READ Procedure	Allows a user to read registered data sets, subject to access restrictions imposed on data sets at registration.
GRANT_REGISTER Procedure	Allows a user to register a data set for remote access.
REVOKE_AUTHORIZE Procedure	Allows the ADMIN user to revoke a user's permission to invoke `DBMS_CLOUD_LINK.GRANT_AUTHORIZATION` and `DBMS_CLOUD_LINK.REVOKE_AUTHORIZATION` procedures.
REVOKE_READ Procedure	Disallows a user from accessing registered data sets of the Autonomous Database instance.
REVOKE_REGISTER Procedure	Disallows a user from registering data sets for remote access. Data sets that were already registered by the user are unaffected.

- GRANT_AUTHORIZE Procedure
- GRANT_READ Procedure
- GRANT_REGISTER Procedure
- REVOKE_AUTHORIZE Procedure
- REVOKE_READ Procedure
- REVOKE_REGISTER Procedure

GRANT_AUTHORIZE Procedure

The procedure grants a user permission to invoke the `DBMS_CLOUD_LINK.GRANT_AUTHORIZATION` and `DBMS_CLOUD_LINK.REVOKE_AUTHORIZATION` procedures.

Syntax

```
DBMS_CLOUD_LINK_ADMIN.GRANT_AUTHORIZE(
      username        IN   VARCHAR2
);
```

Parameters

Parameter	Description
`username`	Specifies a username.

Usage Notes

- To enable authorization for a data set with `DBMS_CLOUD_LINK.GRANT_AUTHORIZATION`, you have to have granted the privilege with `DBMS_CLOUD_LINK_ADMIN.GRANT_AUTHORIZE`. This is true for ADMIN user as well; however ADMIN user can grant this privilege to themself.

GRANT_READ Procedure

The procedure allows a user to read registered data sets, subject to the access restrictions imposed on data sets when a data set is registered using `DBMS_CLOUD_LINK_ADMIN.GRANT_REGISTER`.

Syntax

```
DBMS_CLOUD_LINK_ADMIN.GRANT_READ(
      username         IN   VARCHAR2
);
```

Parameters

Parameter	Description
`username`	Specifies a username.

Usage Notes

- To read data sets, you have to have granted the privilege with `DBMS_CLOUD_LINK_ADMIN.GRANT_READ`. This is true for ADMIN user as well; however ADMIN user can grant this privilege to themself.
- A user can query `SYS_CONTEXT('USERENV', 'CLOUD_LINK_READ_ENABLED')` to check if they are enabled for `READ` access to a data set.

  For example, the following query:

  ```
  SELECT SYS_CONTEXT('USERENV', 'CLOUD_LINK_READ_ENABLED') FROM DUAL;
  ```

  Returns '`YES`' or '`NO`'.

GRANT_REGISTER Procedure

The procedure allows a user to register a data set for remote access.

Syntax

```
DBMS_CLOUD_LINK_ADMIN.GRANT_REGISTER(
      username         IN   VARCHAR2,
      scope            IN   CLOB
);
```

Parameters

Parameter	Description
`username`	Specifies a user name.

Parameter	Description
scope	Specifies the scope in which permissions to publish are to be granted to the specified user. Valid values are: • 'MY$REGION' • 'MY$TENANCY' • 'MY$COMPARTMENT'

Usage Notes

- To register data sets, you have to have granted the privilege with DBMS_CLOUD_LINK_ADMIN.GRANT_REGISTER. This is true for ADMIN user as well; however ADMIN user can grant this privilege to themself.
- A user can query SYS_CONTEXT('USERENV', 'CLOUD_LINK_REGISTER_ENABLED') to check if they are enabled for registering data sets.

  For example, the following query:

  ```
  SELECT SYS_CONTEXT('USERENV', 'CLOUD_LINK_REGISTER_ENABLED') FROM DUAL;
  ```

  Returns 'YES' or 'NO'.

REVOKE_AUTHORIZE Procedure

This procedure disallows a user from invoking DBMS_CLOUD_LINK.GRANT_AUTHORIZATION and DBMS_CLOUD_LINK.REVOKE_AUTHORIZATION procedures.

Syntax

```
DBMS_CLOUD_LINK_ADMIN.REVOKE_AUTHORIZE(
      username        IN   VARCHAR2
);
```

Parameters

Parameter	Description
username	Specifies a username.

REVOKE_READ Procedure

This procedure disallows a user from accessing registered data sets on the Autonomous Database instance.

Syntax

```
DBMS_CLOUD_LINK_ADMIN.REVOKE_READ(
      username        IN   VARCHAR2
);
```

Parameters

Parameter	Description
username	Specifies a username.

Usage Note

- A user can query `SYS_CONTEXT('USERENV', 'CLOUD_LINK_READ_ENABLED')` to check if they are enabled for `READ` access to a data set.

  For example, the following query:

  ```
  SELECT SYS_CONTEXT('USERENV', 'CLOUD_LINK_READ_ENABLED') FROM DUAL;
  ```

  Returns '`YES`' or '`NO`'.

REVOKE_REGISTER Procedure

The procedure disallows a user from registering data sets for remote access. Data sets that were already registered by the user are unaffected.

Syntax

```
DBMS_CLOUD_LINK_ADMIN.REVOKE_REGISTER(
      username         IN   VARCHAR2
);
```

Parameters

Parameter	Description
username	Specifies a user name.

Usage Note

- A user can query `SYS_CONTEXT('USERENV', 'CLOUD_LINK_REGISTER_ENABLED')` to check if they are enabled for registering data sets.

  For example, the following query:

  ```
  SELECT SYS_CONTEXT('USERENV', 'CLOUD_LINK_REGISTER_ENABLED') FROM DUAL;
  ```

  Returns '`YES`' or '`NO`'.

DBMS_CLOUD_MACADM Package

This section covers the `DBMS_CLOUD_MACADM` subprograms provided with Autonomous Database.

- CONFIGURE_DATABASE_VAULT Procedure
  This procedure configures the initial two Oracle Database user accounts, which are granted the `DV_OWNER` and `DV_ACCTMGR` roles, respectively for Autonomous Database.

- DISABLE_DATABASE_VAULT Procedure
  This procedure disables Oracle Database Vault on Autonomous Database. To use this procedure you must have the DV_OWNER role.
- DISABLE_USERMGMT_DATABASE_VAULT Procedure
  This procedure disallows user management related operations for specified components on Autonomous Database with Oracle Database Vault enabled.
- ENABLE_DATABASE_VAULT Procedure
  This procedure enables Oracle Database Vault on Autonomous Database. To use this procedure you must have the DV_OWNER role.
- ENABLE_USERMGMT_DATABASE_VAULT Procedure
  This procedure allows user management with Oracle Database Vault enabled for specified components on Autonomous Database.

CONFIGURE_DATABASE_VAULT Procedure

This procedure configures the initial two Oracle Database user accounts, which are granted the DV_OWNER and DV_ACCTMGR roles, respectively for Autonomous Database.

Syntax

```
DBMS_CLOUD_MACADM.CONFIGURE_DATABASE_VAULT(
       dvowner_uname     IN VARCHAR2,
       dvacctmgr_uname   IN VARCHAR2);
```

Parameters

Parameter	Description
dvowner_uname	Name of the user who will be the Database Vault Owner. This user will be granted the DV_OWNER role.
dvacctmgr_uname	Name of the user who will be the Database Vault Account Manager. This user will be granted the DV_ACCTMGR role. If you omit this setting, the user specified by the dvowner_uname parameter is made the Database Vault Account Manager and granted the DV_ACCTMGR role.

Usage Notes

- Only the ADMIN user can run the DBMS_CLOUD_MACADM.CONFIGURE_DATABASE_VAULT procedure.
- The DBMS_CLOUD_MACADM.CONFIGURE_DATABASE_VAULT procedure does not allow the ADMIN user to be specified as an input for the dvowner_uname or dvacctmgr_uname arguments.

Example

```
BEGIN
   DBMS_CLOUD_MACADM.CONFIGURE_DATABASE_VAULT(
       dvowner_uname         => 'adb_dbv_owner',
       dvacctmgr_uname       => 'adb_dbv_acctmgr');
END;
/
```

DISABLE_DATABASE_VAULT Procedure

This procedure disables Oracle Database Vault on Autonomous Database. To use this procedure you must have the `DV_OWNER` role.

Syntax

```
DBMS_CLOUD_MACADM.DISABLE_DATABASE_VAULT;
```

Usage Notes

After you run `DBMS_CLOUD_MACADM.DISABLE_DATABASE_VAULT` you must restart the Autonomous Database instance.

To use this procedure you must have the `DV_OWNER` role.

Example

```
EXEC DBMS_CLOUD_MACADM.DISABLE_DATABASE_VAULT;
```

DISABLE_USERMGMT_DATABASE_VAULT Procedure

This procedure disallows user management related operations for specified components on Autonomous Database with Oracle Database Vault enabled.

Syntax

```
DBMS_CLOUD_MACADM.DISABLE_USERMGMT_DATABASE_VAULT('component_name');
```

Where: *component_name* is the component name. Valid value is: `APEX`.

APEX is the Oracle APEX component.

Usage Notes

If you enable Oracle Database Vault with Autonomous Database and you want to enforce strict separation of duty to disallow user management related operations for the APEX, use the `DBMS_CLOUD_MACADM.DISABLE_USERMGMT_DATABASE_VAULT` procedure.

To use this procedure you must have the `DV_ACCTMGR` and `DV_ADMIN` roles.

Example

The following example disables user management for the APEX component:

```
EXEC DBMS_CLOUD_MACADM.DISABLE_USERMGMT_DATABASE_VAULT('APEX');
```

ENABLE_DATABASE_VAULT Procedure

This procedure enables Oracle Database Vault on Autonomous Database. To use this procedure you must have the `DV_OWNER` role.

Syntax

```
DBMS_CLOUD_MACADM.ENABLE_DATABASE_VAULT;
```

Usage Notes

After you run `DBMS_CLOUD_MACADM.ENABLE_DATABASE_VAULT` you must restart the Autonomous Database instance.

To use this procedure you must have the `DV_OWNER` role.

Example

The following example enables Oracle Database Vault:

```
BEGIN
   DBMS_CLOUD_MACADM.ENABLE_DATABASE_VAULT;
END;
/
```

ENABLE_USERMGMT_DATABASE_VAULT Procedure

This procedure allows user management with Oracle Database Vault enabled for specified components on Autonomous Database.

Syntax

```
DBMS_CLOUD_MACADM.ENABLE_USERMGMT_DATABASE_VAULT('component_name');
```

Where: *component_name* is the component name. Valid value is: `APEX`.

`APEX` is the Oracle APEX component.

Usage Notes

To use this procedure you must have the `DV_ACCTMGR` and `DV_ADMIN` roles.

Example

The following example enables user management for the APEX component:

```
EXEC DBMS_CLOUD_MACADM.ENABLE_USERMGMT_DATABASE_VAULT('APEX');
```

DBMS_CLOUD_NOTIFICATION Package

The `DBMS_CLOUD_NOTIFICATION` package allows you to send messages or the output of a SQL query to a provider.

- DBMS_CLOUD_NOTIFICATION Overview
- Summary of DBMS_CLOUD_NOTIFICATION Subprograms
  This table summarizes the subprograms included in the package.

DBMS_CLOUD_NOTIFICATION Overview

This scenario describes setup and use of the `DBMS_CLOUD_NOTIFICATION` package.

The supported providers are: Email, Microsoft Teams, and Slack.

Summary of DBMS_CLOUD_NOTIFICATION Subprograms

This table summarizes the subprograms included in the package.

Subprogram	Description
SEND_DATA Procedure	Send SQL query output to a provider.
SEND_MESSAGE Procedure	Send a text message to a provider.

- SEND_DATA Procedure
  The `SEND_DATA` procedure sends the results of the specified query to a provider.
- SEND_MESSAGE Procedure
  The procedure sends a text message to a provider.

SEND_DATA Procedure

The `SEND_DATA` procedure sends the results of the specified query to a provider.

Syntax

```
DBMS_CLOUD_NOTIFICATION.SEND_DATA(
       provider           IN  VARCHAR2,
       credential_name    IN  VARCHAR2,
       query              IN  CLOB,
       params             IN  CLOB
 );
```

Parameters

Parameter	Description
`provider`	Specifies the provider. Valid values are: '`email`', '`msteams`', and '`slack`' This parameter is mandatory.
`credential_name`	The name of the credential object to access the provider. For the email provider, the credential is the name of the credential of the SMTP approved sender that contains its username and password. For the msteams provider, the credential is the name of the credential. For the Slack provider, the credential's username must be "`SLACK_TOKEN`" and the `password` is the Slack Bot Token. This parameter is mandatory.
`query`	Specifies the SQL query to run. Results are sent to the provider. This parameter is mandatory.

Parameter	Description
params	Specifies specific parameters for the provider in JSON format. For the provider type email, the valid parameters are: • sender. This specifies the Email ID of the approved sender in a String value. • smtp_host. This specifies the SMTP host name in a String value. • subject. This specifies the subject of the email in a String value The maximum size is 100 characters. • recipient. This specifies the email IDs of recipients in a String value. Use a comma between email IDs when there are multiple recipients. • to_cc. This specifies the email IDs that are receiving a CC of the email. It is a String value. Use a comma between email IDs when there are multiple CC recipients. • to_bcc. This specifies the email IDs that are receiving a BCC of the email. It is a String value. Use a comma between email IDs when there are multiple BCC recipients. • message. This specifies the message text in a String value. • type. This specifies the output format in a String value as either CSV or JSON. • title. This specifies the title of the attachment of SQL output in a String value. The title should only contain letters, digits, underscores, hyphens, or dots as characters in its value due to it being used to generate a file name. For the provider type msteams, the valid parameters are: • tenant. This specifies the tenant ID in a String format. • team. This specifies the team ID in a String format. • channel. This specifies the channel ID in a String format. • title. This specifies the title of the file in a String format. The maximum size is 50 characters. • type. This specifies the output format in a String value. Valid values are CSV or JSON. For the provider type slack, the valid parameters are: • channel. This specifies the Channel ID in a String value. The Channel ID is a unique ID for a channel and is different from the channel name. In Slack, when you view channel details you can find the Channel ID on the “About” tab. • type. This specifies the output format in a String value. Valid values are CSV or JSON. This parameter is mandatory.

Usage Notes

- Use the procedure DBMS_CLOUD.CREATE_CREDENTIAL(credential_name, username, password) to create the credential object.

  See CREATE_CREDENTIAL Procedure for more information.

- For the email provider, note the following:
  - The user requires an SMTP connection endpoint for the Email Delivery server to obtain smtp_host. The user also requires an SMTP approved sender and its credentials to authenticate with the Email Delivery server to obtain the credential_name. The SMTP connection must be configured and the SMTP credentials must be generated and approved.

  - The maximum message size supported when using `DBMS_CLOUD_NOTIFICATION.SEND_DATA` with the `email` provider is 32k bytes.
  - You can only use `DBMS_CLOUD_NOTIFICATION` for mail notifications with Autonomous Database version 19.21 and above.

- For the `msteams` provider, the user requires the Microsoft Teams App and a bot configured in it. The app needs to be published to the org and installed after obtaining approval from the admin from the admin center. The user also requires access for `Files.ReadWrite.All` and `ChannelSettings.Read.All` permission for Graph API from their Azure Portal. To generate the required token, the user requires `bot_id` in the username and `bot_secret` in the password.

  The maximum file size supported when using `DBMS_CLOUD_NOTIFICATION.SEND_DATA` for Microsoft Teams is 4MB.

- For the `slack` provider, the `username` value can be any valid string and the `password` is the Slack bot token.

Examples

Send SQL output with the `email` provider:

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'EMAIL_CRED',
    username        => 'username',
    password        => 'password');
END;
/

BEGIN
   DBMS_CLOUD_NOTIFICATION.SEND_DATA(
        provider        => 'email',
        credential_name => 'EMAIL_CRED',
        query           => 'SELECT tablespace_name FROM dba_tablespaces',
        params          => json_object('recipient' value  'mark@oracle.com,
suresh@oracle.com',
                                       'to_cc'  value 'nicole@oracle.com1,
jordan@oracle.com',
                                       'to_bcc' value 'manisha@oracle.com',
                                       'subject' value 'Test subject',
                                       'type' value 'json',
                                       'title' value 'mytitle',
                                       'message' value 'This is the message',
                                       'smtp_host' value
'smtp.email.example.com',
                                       'sender'    value
'approver_sender@oracle.com' )
   );
   END;
/
```

Send SQL output with the `msteams` provider:

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(credential_name => 'TEAMS_CRED',
```

```
        username        => 'bot_id',
        password        => 'bot_secret');
END;
/

BEGIN
     DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(provider => 'msteams',
        credential_name => 'TEAMS_CRED',
        query           => 'SELECT tablespace_name FROM dba_tablespaces',
        params          => json_object('tenant'value '5b743bc******c0286',
                                       'team'value '0ae401*********5d2bd',
                                       'channel'value
'19%3a94be023*****%40thread.tacv2',
                                       'title'value 'today',
                                       'type'value 'csv'));
END;
/
```

Send SQL output with the `slack` provider:

```
BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
     credential_name => 'SLACK_CRED',
     username        => 'SLACK_TOKEN',
     password        => 'password');
END;
/
BEGIN
  DBMS_CLOUD_NOTIFICATION.SEND_DATA(
      provider        => 'slack',
      credential_name => 'SLACK_CRED',
      query           => 'SELECT username, account_status, expiry_date FROM
USER_USERS WHERE rownum < 5',
      params          => json_object('channel' value 'C0....08','type' value
'csv'));
END;
/
```

SEND_MESSAGE Procedure

The procedure sends a text message to a provider.

Syntax

```
DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
       provider           IN  VARCHAR2,
       credential_name    IN  VARCHAR2,
       message            IN  CLOB,
       params             IN  CLOB
 );
```

Parameters

Parameter	Description
`provider`	Specifies the provider. Valid values are: '`email`', '`msteams`', and '`slack`' This parameter is mandatory.
`credential_name`	The name of the credential object to access the provider. For the email provider, the credential is the name of the credential of the SMTP approved sender that contains its username and password. For the msteams provider, the credential must contain the `bot_id` and the `bot_secret` key in both the username and the password. For the Slack provider, the credential's username must be "`SLACK_TOKEN`" and the `password` is the Slack Bot Token. This parameter is mandatory.
`message`	Specifies the message text. This parameter is mandatory.
`params`	Specifies specific parameters for the `provider` in JSON format. For the provider type `email`, the valid parameters are: • `sender`. This specifies the Email ID of the approved sender in a `String` value. • `smtp_host`. This specifies the SMTP host name in a `String` value. • `subject`. This specifies the subject of the email in a `String` value. The maximum size is 100 characters. • `recipient`. This specifies the email IDs of recipients in a `String` value. Use a comma between email IDs when there are multiple recipients. • `to_cc`. This specifies the email IDs that are receiving a CC of the email. Use a comma between email IDs when there are multiple CC recipients. • `to_bcc`. This specifies the email IDs that are receiving a BCC of the email. Use a comma between email IDs when there are multiple BCC recipients. For the provider type `msteams`, the valid parameter is: • `channel`. This specifies the Channel ID in a `String` value. For the provider type `slack`, the valid parameter is: • `channel`. This specifies the Channel ID in a `String` value. The Channel ID is a unique ID for a channel and is different from the channel name. In Slack, when you view channel details you can find the Channel ID on the “About” tab. This parameter is mandatory.

Usage Notes

Use the procedure `DBMS_CLOUD.CREATE_CREDENTIAL(credential_name, username, password)` to create the credential object. See CREATE_CREDENTIAL Procedure for more information.

- For the `email` provider, the user requires an SMTP connection endpoint for the Email Delivery server to obtain `smtp_host`. The user also requires an SMTP approved sender and its credentials to authenticate with the Email Delivery server to obtain the `credential_name`. SMTP connection must be configured, and SMTP credentials must be generated and approved.
- For the `msteams` provider, the user requires the Microsoft Teams App and a bot configured in it. The app needs to be published to the org and installed after obtaining approval from

the admin from the admin center. The user also requires access for `Files.ReadWrite.All` and `ChannelSettings.Read.All` permission for Graph API from their Azure Portal. To generate the required token, the user requires `bot_id` in the username and `bot_secret` in the password.

- For the `slack` provider, the `username` value can be any valid string and the `password` is the Slack bot token.

Examples

Send a text message with the `email` provider:

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'EMAIL_CRED',
    username        => 'username',
    password        => 'password');
END;
/

BEGIN
   DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
        provider        => 'email',
        credential_name => 'EMAIL_CRED',
        message         => 'Subject content',
        params          => json_object('recipient' value  'mark@oracle.com,
suresh@oracle.com',
                                       'to_cc'  value 'nicole@oracle.com,
jordan@oracle.com',
                                       'to_bcc' value 'manisha@oracle.com',
                                       'subject' value 'Test subject',
                                       'smtp_host' value
'smtp.email.example.com',
                                       'sender'    value
'approver_sender@oracle.com' )
   );
   END;
/
```

Send a text message with the `msteams` provider:

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(credential_name => 'TEAMS_CRED',
        username        => 'bot_id',
        password        => 'bot_secret');
END;
/

BEGIN
     DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
        provider        => 'msteams',
        credential_name => 'TEAMS_CRED',
        message         => 'text from new teams api',
        params          => json_object('channel'value
'C0....08'),'region'value 'india');
END;
```

```
/
```

Send a text message with the `slack` provider:

```
BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
     credential_name => 'SLACK_CRED',
     username        => 'SLACK_TOKEN',
     password        => 'password');
END;
/
BEGIN
   DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
      provider          => 'slack',
      credential_name   => 'SLACK_CRED',
      message           => 'Send text from Autonomous Database.',
      params            => json_object('channel' value 'C0....08'));
END;
/
```

DBMS_DATA_ACCESS Package

The `DBMS_DATA_ACCESS` package provides routines to generate and manage Pre-Authenticated Request (PAR) URLs for data sets.

- DBMS_DATA_ACCESS Overview
  Describes the use of the `DBMS_DATA_ACCESS` package.
- DBMS_DATA_ACCESS Security Model
  Security on this package can be controlled by granting `EXECUTE` on this package to selected users or roles.
- Summary of DBMS_DATA_ACCESS Subprograms
  This section covers the `DBMS_DATA_ACCESS` subprograms provided with Autonomous Database.

DBMS_DATA_ACCESS Overview

Describes the use of the `DBMS_DATA_ACCESS` package.

`DBMS_DATA_ACCESS` supports these operations:

- Generation of a PAR URL
- Manual invalidation of a PAR URL
- Listing of active PAR URLs

DBMS_DATA_ACCESS Security Model

Security on this package can be controlled by granting `EXECUTE` on this package to selected users or roles.

When a user has been granted `EXECUTE` on `DBMS_DATA_ACCESS` they are able to create, list or invalidate the PAR URLs that are created by the user. In addition, by default the ADMIN user has the following privileges:

- The ADMIN user with `PDB_DBA` role has `EXECUTE` privilege on `DBMS_DATA_ACCESS`.
- The ADMIN user with the `PDB_DBA` role is able to list or invalidate any PAR URL in an Autonomous Database instance.

Summary of DBMS_DATA_ACCESS Subprograms

This section covers the `DBMS_DATA_ACCESS` subprograms provided with Autonomous Database.

Subprogram	Description
GET_PREAUTHENTICATED_URL Procedure	This procedure generates a PAR URL.
INVALIDATE_URL Procedure	This procedure invalidates a PAR URL.
LIST_ACTIVE_URLS Function	This function lists all the currently active PAR URLs.

- GET_PREAUTHENTICATED_URL Procedure
- INVALIDATE_URL Procedure
  This procedure invalidates a PAR URL.
- LIST_ACTIVE_URLS Function
  This function lists all the currently active PAR URLs.

GET_PREAUTHENTICATED_URL Procedure

This procedure generates a PAR URL.

There are two forms, one to generate the PAR URL for a specific object (table or view). The overloaded form, using the `sql_statement` parameter, generates a PAR URL for a SQL statement.

Syntax

```
DBMS_DATA_ACCESS.GET_PREAUTHENTICATED_URL(
    schema_name           IN VARCHAR2,
    schema_object_name    IN VARCHAR2,
    application_user_id   IN VARCHAR2,
    expiration_minutes    IN NUMBER,
    expiration_count      IN NUMBER,
    result               OUT CLOB);

DBMS_DATA_ACCESS.GET_PREAUTHENTICATED_URL(
    sql_statement          IN CLOB,
    application_user_id    IN VARCHAR2,
    expiration_minutes     IN NUMBER,
    expiration_count       IN NUMBER,
    result                OUT CLOB);
```

Parameters

Parameter	Description
`schema_name`	Specifies the owner of the object.
`schema_object_name`	Specifies the schema object (table or view).

Parameter	Description
sql_statement	Specifies the SELECT statement query text. PAR URLs do not accept bind variables.
application_user_id	Specifies an application user ID value. When the PAR URL is accessed, the value of application_user_id specified during PAR URL generation is available through: sys_context('DATA_ACCESS_CONTEXT$', 'USER_IDENTITY') You can define VPD Policies that make use of this value in the Application Context to restrict the rows visible to the application user.
expiration_minutes	Duration in minutes of validity of PAR URL. The maximum allowed expiration time is 90 days (129600 minutes). If the value is set to greater than 129600, the value used is 129600 minutes (90 days). If expiration_minutes is specified as a non-null value, expiration_count must not be set to a non-null value. Both cannot be non-null at the same time. Default value: when expiration_minutes is not provided or when expiration_minutes is provided as NULL, the value is set to 90 days (129600 minutes).
expiration_count	Number of accesses allowed on the PAR URL. There is no default value. If expiration_count is not specified and expiration_minutes is not specified, expiration_minutes is set to 90 days (129600 minutes). If expiration_count is specified as a non-null value, expiration_minutes must not be set to a non-null value. Both cannot be non-null at the same time.
result	(CLOB)

Usage Note

- There is a limit of 128 active PAR URLs on an Autonomous Database instance.

Examples

```
DECLARE
   status CLOB;
   BEGIN
   DBMS_DATA_ACCESS.GET_PREAUTHENTICATED_URL(
      schema_name => 'USER1',
      schema_object_name => 'STUDENTS_VIEW',
      expiration_minutes => 120,
      result => status);
   dbms_output.put_line(status);
END;
/


DECLARE
   status CLOB;
   par_url_app_string CLOB;
   BEGIN
       par_url_app_string := 1919292929;
```

```
       DBMS_DATA_ACCESS.GET_PREAUTHENTICATED_URL(
            sql_statement => 'SELECT student_id, student_name FROM
STUDENTS_VIEW ORDER BY student_id',
            application_user_id => par_url_app_string,
            expiration_count => 25,
            result => status);
END;
/
```

INVALIDATE_URL Procedure

This procedure invalidates a PAR URL.

Syntax

```
DBMS_DATA_ACCESS.INVALIDATE_URL(
    id                  IN VARCHAR2,
    kill_sessions       IN BOOLEAN DEFAULT FALSE,
    result              OUT CLOB);
```

Parameters

Parameter	Description
`id`	Specifies the owner of the object.
`kill_sessions`	By default, existing sessions that may be in the middle of accessing data using a PAR URL are not killed. When TRUE, this parameter specifies that such existing sessions should be killed, so that the invalidation does not leave any ongoing access to the data set. Valid values: `TRUE` | `FALSE`.
`result`	Provides JSON to indicate whether invalidation is a success or a failure (`CLOB`).

LIST_ACTIVE_URLS Function

This function lists all the currently active PAR URLs.

Syntax

```
DBMS_DATA_ACCESS.LIST_ACTIVE_URLS RETURN CLOB;
```

Parameters

Parameter	Description
`RETURN`	The return value is a JSON array.

Example

```
DECLARE
   result CLOB;
   BEGIN
       result := DBMS_DATA_ACCESS.LIST_ACTIVE_URLS;
```

```
        DBMS_OUTPUT.PUT_LINE(result);
    END;

[
    {
        "id": "89fa6081-ec6b-4179-9b06-a93af8fbd4b7",
        "schema_name": "SCOTT",
        "schema_object_name": "EMPLOYEE",
        "created_by": "ADMIN",
        "application_user_id": "AMIT",
        "expiration_time": "2023-01-14T23:41:01.029Z",
        "expiration_count": 100,
        "access_count": 9,
        "created": "2023-01-10T19:41:01.285Z"
    },
    {
        "id": "263d2cd7-3bc0-41a7-8cb9-438a2d843481",
        "sql_statement": "select name from v$pdbs",
        "created_by": "ADMIN",
        "application_user_id": "AMIT",
        "expiration_time": "2023-01-15T00:04:30.578Z",
        "expiration_count": 100,
        "access_count": 0,
        "created": "2023-01-10T20:04:30.607Z"
    }
]
```

Usage Note

- The behavior of `DBMS_DATA_ACCESS.LIST_ACTIVE_URLS` is dependent on the invoker. If the invoker is ADMIN or any user with `PDB_DBA` role, the function lists all active PAR URLs, regardless of the user who generated the PAR URL. If the invoker is not the ADMIN user and not a user with `PDB_DBA` role, the list includes only the active PAR URLs generated by the invoker.

DBMS_PIPE Package

The `DBMS_PIPE` package lets two or more sessions in the same instance communicate.

Oracle Autonomous Database supports core `DBMS_PIPE` functionality as available in Oracle Database 19c, plus extensions.

See DBMS_PIPE for details about the core `DBMS_PIPE` functionality provided in Oracle Database.

- DBMS_PIPE Overview for Singleton Pipes
  Pipe functionality has several potential applications: external service interface, debugging, independent transactions, and alerts.
- Summary of DBMS_PIPE Subprograms for Singleton Pipes
  This table lists the `DBMS_PIPE` subprograms and briefly describes them.
- DBMS_PIPE Overview for Persistent Messaging Pipes
  Pipe functionality has several potential applications: external service interface, debugging, independent transactions, and alerts.

- Summary of DBMS_PIPE Subprograms for Persistent Messaging
  This table lists the DBMS_PIPE subprograms and briefly describes them.

DBMS_PIPE Overview for Singleton Pipes

Pipe functionality has several potential applications: external service interface, debugging, independent transactions, and alerts.

On Autonomous Database the DBMS_PIPE package has extended functionality to support singleton pipes.

Singleton pipe features in DBMS_PIPE provide the following:

- Ability to cache and retrieve a custom message, of up to 32,767 bytes, in Oracle database memory. The message size maximum of 32,767 bytes is applicable to all Pipes, including Singleton Pipes. Previous versions of DBMS_PIPE had a smaller maximum message size.
- Share the cached message across multiple database sessions with concurrent reads.
- Cache Invalidation methods:
  - Explicit cache invalidation controlled by user.
  - Cache invalidation after a user specified parameter (shelflife) time interval (in seconds).
- Declarative and easy to use PL/SQL APIs for caching.
- Supports both Read-Only and Read-Write databases.

A Singleton Pipe can be any one of the supported DBMS_PIPE types:

- **Implicit Pipe:** Automatically created when a message is sent with an unknown pipe name using the DBMS_PIPE.SEND_MESSAGE function.
- **Explicit Pipe:** Created using the DBMS_PIPE.CREATE_PIPE function with a user specified pipe name.
- **Public Pipe:** Accessible by any user with EXECUTE permission on DBMS_PIPE package
- **Private Pipe:** Accessible by sessions with the same user as the pipe creator.

Summary of DBMS_PIPE Subprograms for Singleton Pipes

This table lists the DBMS_PIPE subprograms and briefly describes them.

Table 6-1 DBMS_PIPE Package Subprograms

Subprogram	Description
CREATE_PIPE Function	Creates a pipe (necessary for private pipes)
NEXT_ITEM_TYPE Function	Returns datatype of next item in buffer
PACK_MESSAGE Procedures	Builds message in local buffer
PURGE Procedure	Purges contents of named pipe
RECEIVE_MESSAGE Function	Copies message from named pipe into local buffer
RESET_BUFFER Procedure	Purges contents of local buffer
REMOVE_PIPE Function	Removes the named pipe
SEND_MESSAGE Function	Sends message on named pipe: This implicitly creates a public pipe if the named pipe does not exist

Table 6-1 (Cont.) DBMS_PIPE Package Subprograms

Subprogram	Description
UNIQUE_SESSION_NAME Function	Returns unique session name
UNPACK_MESSAGE Procedures	Accesses next item in buffer

- CREATE_PIPE Function
  This function explicitly creates a public or private pipe. If the `private` flag is `TRUE`, then the pipe creator is assigned as the owner of the private pipe.
- RECEIVE_MESSAGE Function
  This function copies the message into the local message buffer.
- SEND_MESSAGE Function
  This function sends a message on the named pipe.

CREATE_PIPE Function

This function explicitly creates a public or private pipe. If the `private` flag is `TRUE`, then the pipe creator is assigned as the owner of the private pipe.

Explicitly-created pipes can only be removed by calling `REMOVE_PIPE`, or by shutting down the instance.

In order to create a Singleton Pipe, set the `singleton` parameter to `TRUE`. The following arguments are applicable to Singleton Pipes:

- `singleton`: Indicates that the pipe should be created as a Singleton Pipe (default value: `FALSE`).
- `shelflife`: Optionally specify a shelflife expiration (in seconds) of cached message in the Singleton Pipe. It can be used for implicit invalidation of message in Singleton Pipe.

  The message `shelflife` in Singleton Pipe can also be specified when you send a message message (see SEND_MESSAGE Function).

Syntax

```
DBMS_PIPE.CREATE_PIPE (
   pipename     IN VARCHAR2,
   maxpipesize  IN INTEGER DEFAULT 66536,
   private      IN BOOLEAN DEFAULT TRUE,
   singleton    IN BOOLEAN DEFAULT FALSE,
   shelflife    IN INTEGER DEFAULT 0)
RETURN INTEGER;
```

Parameters

Table 6-2 CREATE_PIPE Function Parameters

Parameter	Description
pipename	Name of the pipe you are creating. You must use this name when you call SEND_MESSAGE and RECEIVE_MESSAGE. This name must be unique across the instance. Caution: Do not use pipe names beginning with ORA$. These are reserved for use by procedures provided by Oracle. Pipename should not be longer than 128 bytes, and is case insensitive. At this time, the name cannot contain Globalization Support characters.
maxpipesize	The maximum size allowed for the pipe, in bytes. The total size of all of the messages on the pipe cannot exceed this amount. The message is blocked if it exceeds this maximum. The default maxpipesize is 66536 bytes. The maxpipesize for a pipe becomes a part of the characteristics of the pipe and persists for the life of the pipe. Callers of SEND_MESSAGE with larger values cause the maxpipesize to be increased. Callers with a smaller value use the existing, larger value. The default maxpipesize of 65536 is applicable for all Pipes.
private	Uses the default, TRUE, to create a private pipe. Public pipes can be implicitly created when you call SEND_MESSAGE.
singleton	Use TRUE to create a Singleton Pipe. Default value: FALSE
shelflife	Expiration time in seconds of a message cached in Singleton Pipe. After the specified shelflife time is exceeded, the message is no longer accessible from the Pipe. The parameter shelflife is only applicable to a Singleton Pipe. Default value is 0, implying that the message never expires.

Return Values

Table 6-3 CREATE_PIPE Function Return Values

Return	Description
0	Successful. If the pipe already exists and the user attempting to create it is authorized to use it, then Oracle returns 0, indicating success, and any data already in the pipe remains.
6	Failed to convert existing pipe to singleton pipe. • Implicit pipe with more than one existing message cannot be converted to a Singleton Pipe. • For an explicit pipe that is not Singleton, DBMS_PIPE.SEND_MESSAGE cannot send a message with singleton argument set to TRUE.
7	A non-zero value is given for the shelflife parameter and the pipe is not a singleton pipe.

Table 6-3 (Cont.) CREATE_PIPE Function Return Values

Return	Description
ORA-23322	Failure due to naming conflict. If a pipe with the same name exists and was created by a different user, then Oracle signals error ORA-23322, indicating the naming conflict.

Exceptions

Table 6-4 CREATE_PIPE Function Exception

Exception	Description
Null pipe name	Permission error: Pipe with the same name already exists, and you are not allowed to use it.

Example

Create a Singleton Pipe with shelflife of 1 hour.

```
DECLARE
  l_status INTEGER;
BEGIN
  l_status := DBMS_PIPE.create_pipe(pipename  => 'MY_PIPE1',
                                    private   => TRUE,
                                    singleton => TRUE,
                                    shelflife => 3600);
END;
/
```

RECEIVE_MESSAGE Function

This function copies the message into the local message buffer.

Syntax

```
DBMS_PIPE.RECEIVE_MESSAGE (
   pipename      IN VARCHAR2,
   timeout       IN INTEGER      DEFAULT maxwait,
   cache_func    IN VARCHAR2     DEFAULT NULL)
RETURN INTEGER;
```

Parameters

Table 6-5 RECEIVE_MESSAGE Function Parameters

Parameter	Description
pipename	Name of the pipe on which you want to receive a message. Names beginning with ORA$ are reserved for use by Oracle.

Table 6-5 (Cont.) RECEIVE_MESSAGE Function Parameters

Parameter	Description
timeout	Time to wait for a message, in seconds. A timeout of 0 lets you read without blocking. The timeout does not include the time spent in execution cache function specified in the cache_func parameter. Default value: is the constant MAXWAIT, which is defined as 86400000 (1000 days).
cache_func	Cache function name to automatically cache a message in a Singleton Pipe. The name of the function should be fully qualified with the owner schema: • *OWNER.FUNCTION_NAME* • *OWNER.PACKAGE.FUNCTION_NAME* Default value: NULL

Return Values

Table 6-6 RECEIVE_MESSAGE Function Return Values

Return	Description
0	Success
1	Timed out. If the pipe was implicitly-created and is empty, then it is removed.
2	Record in the pipe is too large for the buffer.
3	An interrupt occurred.
8	Cache function can only be specified when using a Singleton Pipe.
ORA-23322	User has insufficient privileges to read from the pipe.

Usage Notes

To receive a message from a pipe, first call RECEIVE_MESSAGE. When you receive a message, it is removed from the pipe; hence, a message can only be received once. For implicitly-created pipes, the pipe is removed after the last record is removed from the pipe.

If the pipe that you specify when you call RECEIVE_MESSAGE does not already exist, then Oracle implicitly creates the pipe and waits to receive the message. If the message does not arrive within a designated timeout interval, then the call returns and the pipe is removed.

After receiving the message, you must make one or more calls to UNPACK_MESSAGE to access the individual items in the message. The UNPACK_MESSAGE procedure is overloaded to unpack items of type DATE, NUMBER, VARCHAR2, and there are two additional procedures to unpack RAW and ROWID items. If you do not know the type of data that you are attempting to unpack, then call NEXT_ITEM_TYPE to determine the type of the next item in the buffer.

Cache Function Parameter

Singleton Pipes support cache function to automatically cache a message in the pipe in case of the following two scenarios:

- Singleton Pipe is empty.
- Message in Singleton Pipe is invalid due to `shelflife` time elapsed.

The name of the function should be fully qualified with the owner schema:

- *OWNER.FUNCTION_NAME*
- *OWNER.PACKAGE.FUNCTION_NAME*

To use a cache function the current session user that invokes `DBMS_PIPE.RECEIVE_MESSAGE` must have required privileges to execute the cache function.

Cache Function Syntax

```
CREATE OR REPLACE FUNCTION cache_function_name (
       pipename  IN VARCHAR2
) RETURN INTEGER;
```

Parameter	Datatype	Description
pipename	VARCHAR2	Name of the Singleton Pipe.

Return	Description
0	Success
Non-zero	Failure value returned from DBMS_PIPE.RECEIVE_MESSAGE

Define a cache function to provide encapsulation and abstraction of complexity from the reader sessions of Singleton Pipe. The typical operations within a cache function would be:

- Create a Singleton Pipe, for an Explicit Pipe, using `DBMS_PIPE.CREATE_PIPE`.
- Create the message to cache in the Singleton Pipe.
- Send Message to Singleton Pipe, optionally specifying a `shelflife` for the implicit message.

Exceptions

Table 6-7 RECEIVE_MESSAGE Function Exceptions

Exception	Description
Null pipe name	Permission error. Insufficient privilege to remove the record from the pipe. The pipe is owned by someone else.

Example

```
DECLARE
  l_status INTEGER;
BEGIN
  l_status := DBMS_PIPE.receive_message(pipename   => 'MY_PIPE1',
                                        timeout    => 1,
                                        cache_func =>
'MY_USER.MY_CACHE_FUNC');
```

```
END;
/
```

SEND_MESSAGE Function

This function sends a message on the named pipe.

The message is contained in the local message buffer, which was filled with calls to `PACK_MESSAGE`. You can create a pipe explicitly using `CREATE_PIPE`, otherwise, it is created implicitly.

To create an implicit Singleton pipe, set the `singleton` parameter to `TRUE`. The following arguments are applicable to Singleton Pipes:

- `singleton`: Indicates that the pipe should be created as a Singleton Pipe (default value: `FALSE`).
- `shelflife`: Optionally specify a shelflife expiration of cached message in the Singleton Pipe. It can be used for implicit invalidation of message in Singleton Pipe.
  This argument is applicable for implicit as well as explicit Singleton pipes. A `shelflife` value specified in SEND_MESSAGE Function overwrites the `shelflife` specified for the Explicit Singleton Pipe in CREATE_PIPE Function and will be the default for any new messages cached in the Singleton Pipe.

Syntax

```
DBMS_PIPE.SEND_MESSAGE (
    pipename     IN VARCHAR2,
    timeout      IN INTEGER DEFAULT MAXWAIT,
    maxpipesize  IN INTEGER DEFAULT 65536,
    singleton    IN BOOLEAN DEFAULT FALSE,
    shelflife    IN INTEGER DEFAULT 0)
RETURN INTEGER;
```

Parameters

Table 6-8 SEND_MESSAGE Function Parameters

Parameter	Description
`pipename`	Name of the pipe on which you want to place the message. If you are using an explicit pipe, then this is the name that you specified when you called `CREATE_PIPE`. Caution: Do not use pipe names beginning with '`ORA$`'. These names are reserved for use by procedures provided by Oracle. Pipename should not be longer than 128 bytes, and is case-insensitive. At this time, the name cannot contain Globalization Support characters.
`timeout`	Time to wait while attempting to place a message on a pipe, in seconds. The default value is the constant `MAXWAIT`, which is defined as 86400000 (1000 days).

Table 6-8 (Cont.) SEND_MESSAGE Function Parameters

Parameter	Description
maxpipesize	Maximum size allowed for the pipe, in bytes. The total size of all the messages on the pipe cannot exceed this amount. The message is blocked if it exceeds this maximum. The default is 65536 bytes. The maxpipesize for a pipe becomes a part of the characteristics of the pipe and persists for the life of the pipe. Callers of SEND_MESSAGE with larger values cause the maxpipesize to be increased. Callers with a smaller value simply use the existing, larger value. Specifying maxpipesize as part of the SEND_MESSAGE procedure eliminates the need for a separate call to open the pipe. If you created the pipe explicitly, then you can use the optional maxpipesize parameter to override the creation pipe size specifications. The default maxpipesize of 65536 is applicable for all Pipes.
singleton	Use TRUE to create a Singleton Pipe. Default value: FALSE
shelflife	Expiration time in seconds of a message cached in Singleton Pipe. After the specified shelflife time is exceeded, the message is no longer accessible from the Pipe. The parameter shelflife is only applicable to a Singleton Pipe. Default value is 0, implying that the message never expires.

Return Values

Table 6-9 SEND_MESSAGE Function Return Values

Return	Description
0	Success. If the pipe already exists and the user attempting to create it is authorized to use it, then Oracle returns 0, indicating success, and any data already in the pipe remains. If a user connected as SYSDBS/SYSOPER re-creates a pipe, then Oracle returns status 0, but the ownership of the pipe remains unchanged.
1	Timed out. This procedure can timeout either because it cannot get a lock on the pipe, or because the pipe remains too full to be used. If the pipe was implicitly-created and is empty, then it is removed.
3	An interrupt occurred. If the pipe was implicitly created and is empty, then it is removed.
6	Failed to convert existing pipe to singleton pipe. • Implicit pipe with more than one existing message cannot be converted to a Singleton Pipe. • For an explicit pipe that is not Singleton, DBMS_PIPE.SEND_MESSAGE cannot send a message with singleton argument set to TRUE.
7	A non-zero value is given for the shelflife parameter and the pipe is not a singleton pipe.

Table 6-9 (Cont.) SEND_MESSAGE Function Return Values

Return	Description
ORA-23322	Insufficient privileges. If a pipe with the same name exists and was created by a different user, then Oracle signals error ORA-23322, indicating the naming conflict.

Exceptions

Table 6-10 SEND_MESSAGE Function Exception

Exception	Description
Null pipe name	Permission error. Insufficient privilege to write to the pipe. The pipe is private and owned by someone else.

DBMS_PIPE Overview for Persistent Messaging Pipes

Pipe functionality has several potential applications: external service interface, debugging, independent transactions, and alerts.

On Autonomous Database the DBMS_PIPE package has extended functionality to support persistent messaging pipes.

Persistent Messages in DBMS_PIPE:

- Support the ability to send and retrieve very large messages.
- Support a large number of pipe messages.
- Support sharing of messages within a single database, across multiple databases, and across databases in different regions.
- Support multiple pipes using the same Cloud Object Store location URI.

  Persistent messaging functionality allows two or more database sessions to communicate with messages that are stored in Cloud Object Store. Using this functionality messages in a pipe can be made available to only the current database or they can be made available to multiple databases in the same region or across different regions.

  A Persistent Messaging Pipe can be any one of the supported DBMS_PIPE types:

  - **Implicit Pipe:** Automatically created when a message is sent with an unknown pipe name using the DBMS_PIPE.SEND_MESSAGE function.
  - **Explicit Pipe:** Created using the DBMS_PIPE.CREATE_PIPE function with a user specified pipe name.
  - **Public Pipe:** Accessible by any user with EXECUTE permission on DBMS_PIPE package.
  - **Private Pipe:** Accessible by sessions with the same user as the pipe creator.

> **Note:**
>
> When send and receive messages across different databases using persistent messages, Oracle recommends you call `DBMS_PIPE.CREATE_PIPE` before sending or receiving messages. Creating an explicit pipe with `DBMS_PIPE.CREATE_PIPE` ensures that a pipe is created with the access permissions you want, either public or private (by setting the `PRIVATE` parameter to `FALSE` or using the default value `TRUE`).

Summary of DBMS_PIPE Subprograms for Persistent Messaging

This table lists the `DBMS_PIPE` subprograms and briefly describes them.

Table 6-11 DBMS_PIPE Package Subprograms

Subprogram	Description
CREATE_PIPE Function	Creates a pipe (necessary for private pipes).
GET_CREDENTIAL_NAME Function	Returns the global `credential_name` variable value.
GET_LOCATION_URI Function	Returns the global `location_uri` variable value that is used as a default location URI for use when a message is stored in Cloud Object Store.
NEXT_ITEM_TYPE Function	Returns datatype of next item in buffer.
PACK_MESSAGE Procedures	Builds message in local buffer.
RECEIVE_MESSAGE Function	Copies message from named pipe into local buffer.
RESET_BUFFER Procedure	Purges contents of local buffer.
REMOVE_PIPE Function	Removes the named pipe.
SEND_MESSAGE Function	Sends message on a named pipe: This implicitly creates a public pipe if the named pipe does not exist.
SET_CREDENTIAL_NAME Procedure	Sets the `credential_name` variable that is used as a default credential for messages that are stored in Cloud Object Store.
SET_LOCATION_URI Procedure	Sets the global `location_uri` variable that is used as a default location URI for messages that are stored in Cloud Object Store.
UNIQUE_SESSION_NAME Function	Returns unique session name.
UNPACK_MESSAGE Procedures	Accesses next item in buffer.

- CREATE_PIPE Function
  This function explicitly creates a public or private pipe. If the `private` flag is `TRUE`, then the pipe creator is assigned as the owner of the private pipe.
- GET_CREDENTIAL_NAME Function
  This function returns the global `credential_name` variable value for use when messages are stored to Cloud Object Store.
- GET_LOCATION_URI Function
  This function returns the global `location_uri` variable value that can be used as a default location URI when pipe messages are stored to Cloud Object Store.
- RECEIVE_MESSAGE Function
  This function copies the message into the local message buffer.

- SEND_MESSAGE Function
  This function sends a message on the named pipe.
- SET_CREDENTIAL_NAME Procedure
  This procedure sets the `credential_name` variable that is used as a default credential when pipe messages are stored in Cloud Object Store.
- SET_LOCATION_URI Procedure
  This procedure sets the global `location_uri` variable.

CREATE_PIPE Function

This function explicitly creates a public or private pipe. If the `private` flag is `TRUE`, then the pipe creator is assigned as the owner of the private pipe.

Explicitly-created pipes can only be removed by calling `REMOVE_PIPE`, or by shutting down the instance.

Syntax

```
DBMS_PIPE.CREATE_PIPE (
   pipename     IN VARCHAR2,
   maxpipesize  IN INTEGER DEFAULT 66536,
   private      IN BOOLEAN DEFAULT TRUE)
RETURN INTEGER;
```

Parameters

Table 6-12 CREATE_PIPE Function Parameters

Parameter	Description
`pipename`	Name of the pipe you are creating. You must use this name when you call `SEND_MESSAGE` and `RECEIVE_MESSAGE`. This name must be unique across the instance. Caution: Do not use pipe names beginning with `ORA$`. These are reserved for use by procedures provided by Oracle. Pipename should not be longer than 128 bytes, and is case insensitive. At this time, the name cannot contain Globalization Support characters.
`maxpipesize`	The maximum size allowed for the pipe, in bytes. The total size of all of the messages on the pipe cannot exceed this amount. The message is blocked if it exceeds this maximum. The default `maxpipesize` is 66536 bytes. The `maxpipesize` for a pipe becomes a part of the characteristics of the pipe and persists for the life of the pipe. Callers of `SEND_MESSAGE` with larger values cause the `maxpipesize` to be increased. Callers with a smaller value use the existing, larger value. The default `maxpipesize` of 65536 is applicable for all Pipes.
`private`	Uses the default, `TRUE`, to create a private pipe. Public pipes can be implicitly created when you call `SEND_MESSAGE`.

Return Values

Table 6-13 CREATE_PIPE Function Return Values

Return	Description
0	Successful. If the pipe already exists and the user attempting to create it is authorized to use it, then Oracle returns 0, indicating success, and any data already in the pipe remains.
ORA-23322	Failure due to naming conflict. If a pipe with the same name exists and was created by a different user, then Oracle signals error ORA-23322, indicating the naming conflict.

Exceptions

Table 6-14 CREATE_PIPE Function Exception

Exception	Description
Null pipe name	Permission error: Pipe with the same name already exists, and you are not allowed to use it.

Example

Create an explicit private named MY_PIPE1

```
DECLARE
  l_status INTEGER;
BEGIN
  l_status := DBMS_PIPE.create_pipe(
      pipename  => 'MY_PIPE1',
      private   => TRUE);
END;
/
```

GET_CREDENTIAL_NAME Function

This function returns the global credential_name variable value for use when messages are stored to Cloud Object Store.

Syntax

```
DBMS_PIPE.GET_CREDENTIAL_NAME
         RETURN VARCHAR2;
```

Return Values

Return Value	Description
credential_name	The name of the credential to access the Cloud Object Storage.

Example

```
DECLARE
  credential_name     VARCHAR2(400)
BEGIN
  credential_name := DBMS_PIPE.GET_CREDENTIAL_NAME;
END;
/
```

GET_LOCATION_URI Function

This function returns the global `location_uri` variable value that can be used as a default location URI when pipe messages are stored to Cloud Object Store.

Syntax

```
DBMS_PIPE.GET_LOCATION_URI
        RETURN VARCHAR2;
```

Return Value

Return Value	Description
`location_uri`	The object URI.

Example

```
DECLARE
  location_uri     VARCHAR2(400)
BEGIN
  location_uri := DBMS_PIPE.GET_LOCATION_URI;
END;
/
```

RECEIVE_MESSAGE Function

This function copies the message into the local message buffer.

Syntax

```
DBMS_PIPE.RECEIVE_MESSAGE (
   pipename          IN VARCHAR2,
   timeout           IN INTEGER  DEFAULT maxwait,
   credential_name   IN VARCHAR2 DEFAULT null,
   location_uri      IN VARCHAR2)
RETURN INTEGER;
```

Parameters

Table 6-15 RECEIVE_MESSAGE Function Parameters

Parameter	Description
pipename	Name of the pipe on which you want to receive a message. Names beginning with ORA$ are reserved for use by Oracle.
timeout	Time to wait for a message, in seconds. A timeout of 0 lets you read without blocking. The timeout does not include the time spent running the cache function specified with the cache_func parameter. Default value: is the constant MAXWAIT, which is defined as 86400000 (1000 days).
credential_name	The credential name for the cloud store used to store messages. The credential_name is a package argument that is by default initialized as NULL. You can set this value before calling DBMS_PIPE.RECEIVE_MESSAGE. The passed parameter value takes precedence over the global variable's value. The credential object must have EXECUTE and READ/WRITE privileges by the user running DBMS_PIPE.RECEIVE_MESSAGE. The credential_name value can be an OCI resource principal, Azure service principal, Amazon Resource Name (ARN), or a Google service account. See Configure Policies and Roles to Access Resources for more information on resource principal based authentication.
location_uri	The location URI for the cloud store used to store messages. The location_uri is a global variable that by default is initialized as NULL. You can set this value before calling DBMS_PIPE.RECEIVE_MESSAGE. The passed parameter value takes precedence over the global variable's value.

Return Values

Table 6-16 RECEIVE_MESSAGE Function Return Values

Return	Description
0	Success
1	Timed out. If the pipe was implicitly-created and is empty, then it is removed.
2	Record in the pipe is too large for the buffer.
3	An interrupt occurred.
ORA-23322	User has insufficient privileges to read from the pipe.

Usage Notes

- To receive a message from a pipe, first call RECEIVE_MESSAGE. When you receive a message, it is removed from the pipe; hence, a message can only be received once. For implicitly-created pipes, the pipe is removed after the last record is removed from the pipe.

- If the pipe that you specify when you call `RECEIVE_MESSAGE` does not already exist, then Oracle implicitly creates the pipe and waits to receive the message. If the message does not arrive within a designated timeout interval, then the call returns and the pipe is removed.
- After receiving the message, you must make one or more calls to `UNPACK_MESSAGE` to access the individual items in the message. The `UNPACK_MESSAGE` procedure is overloaded to unpack items of type `DATE`, `NUMBER`, `VARCHAR2`, and there are two additional procedures to unpack `RAW` and `ROWID` items. If you do not know the type of data that you are attempting to unpack, then call `NEXT_ITEM_TYPE` to determine the type of the next item in the buffer.
- Persistent messages are guaranteed to either be written or read by exactly one process. This prevents message content inconsistency due to concurrent writes and reads. Using a persistent messaging pipe, `DBMS_PIPE` allows only one operation, sending a message or a receiving message to be active at a given time. However, if an operation is not possible due to an ongoing operation, the process retries periodically until the `timeout` value is reached.
- If you use Oracle Cloud Infrastructure Object Storage to store messages, you can use Oracle Cloud Infrastructure Native URIs or Swift URIs. However, the location URI and the credential must match in type as follows:
  - If you use a native URI format to access Oracle Cloud Infrastructure Object Storage, you must use Native Oracle Cloud Infrastructure Signing Keys authentication in the credential object.
  - If you use Swift URI format to access Oracle Cloud Infrastructure Object Storage, you must use an auth token authentication in the credential object.

Exceptions

Table 6-17 RECEIVE_MESSAGE Function Exceptions

Exception	Description
`Null pipe name`	Permission error. Insufficient privilege to remove the record from the pipe. The pipe is owned by someone else.

SEND_MESSAGE Function

This function sends a message on the named pipe.

The message is contained in the local message buffer, which was filled with calls to `PACK_MESSAGE`. You can create a pipe explicitly using `CREATE_PIPE`, otherwise, it is created implicitly.

Syntax

```
DBMS_PIPE.SEND_MESSAGE (
    pipename          IN VARCHAR2,
    timeout           IN INTEGER DEFAULT MAXWAIT,
    credential_name   IN VARCHAR2 DEFAULT null,
    location_uri      IN VARCHAR2 )
RETURN INTEGER;
```

Parameters

Table 6-18 SEND_MESSAGE Function Parameters

Parameter	Description
credential_name	The credential name for the cloud store used to store messages. The credential_name is a package argument that is by default initialized as NULL. You can set this value before calling DBMS_PIPE.SEND_MESSAGE. The passed parameter value takes precedence over the global variable's value. The credential object must have EXECUTE and READ/WRITE privileges by the user running DBMS_PIPE.SEND_MESSAGE. The credential_name value can be an OCI resource principal, Azure service principal, Amazon Resource Name (ARN), or a Google service account. See Configure Policies and Roles to Access Resources for more information on resource principal based authentication.
location_uri	The location URI for the cloud store used to store messages. The location_uri is a global variable that by default is initialized as NULL. You can set this value before calling DBMS_PIPE.SEND_MESSAGE. The passed parameter value takes precedence over the global variable's value.
maxpipesize	Maximum size allowed for the pipe, in bytes. The total size of all the messages on the pipe cannot exceed this amount. The message is blocked if it exceeds this maximum. The default is 65536 bytes. The maxpipesize for a pipe becomes a part of the characteristics of the pipe and persists for the life of the pipe. Callers of SEND_MESSAGE with larger values cause the maxpipesize to be increased. Callers with a smaller value simply use the existing, larger value. Specifying maxpipesize as part of the SEND_MESSAGE procedure eliminates the need for a separate call to open the pipe. If you created the pipe explicitly, then you can use the optional maxpipesize parameter to override the creation pipe size specifications. The default maxpipesize of 65536 is applicable for all Pipes.
pipename	Name of the pipe on which you want to place the message. If you are using an explicit pipe, then this is the name that you specified when you called CREATE_PIPE. Caution: Do not use pipe names beginning with 'ORA$'. These names are reserved for use by procedures provided by Oracle. Pipename should not be longer than 128 bytes, and is case-insensitive. At this time, the name cannot contain Globalization Support characters.
timeout	Time to wait while attempting to place a message on a pipe, in seconds. The default value is the constant MAXWAIT, which is defined as 86400000 (1000 days).

Return Values

Table 6-19 SEND_MESSAGE Function Return Values

Return	Description
0	Success. If the pipe already exists and the user attempting to create it is authorized to use it, then Oracle returns 0, indicating success, and any data already in the pipe remains. If a user connected as SYSDBS/SYSOPER re-creates a pipe, then Oracle returns status 0, but the ownership of the pipe remains unchanged.
1	Timed out. This procedure can timeout either because it cannot get a lock on the pipe, or because the pipe remains too full to be used. If the pipe was implicitly-created and is empty, then it is removed.
3	An interrupt occurred. If the pipe was implicitly created and is empty, then it is removed.
ORA-23322	Insufficient privileges. If a pipe with the same name exists and was created by a different user, then Oracle signals error ORA-23322, indicating the naming conflict.

Usage Notes

- Persistent messages are guaranteed to either be written or read by exactly one process. This prevents message content inconsistency due to concurrent writes and reads. Using a persistent messaging pipe, DBMS_PIPE allows only one operation, sending a message or a receiving message to be active at a given time. However, if an operation is not possible due to an ongoing operation, the process retries periodically until the timeout value is reached.
- If you use Oracle Cloud Infrastructure Object Storage to store messages, you can use Oracle Cloud Infrastructure Native URIs or Swift URIs. However, the location URI and the credential must match in type as follows:
  - If you use a native URI format to access Oracle Cloud Infrastructure Object Storage, you must use Native Oracle Cloud Infrastructure Signing Keys authentication in the credential object.
  - If you use Swift URI format to access Oracle Cloud Infrastructure Object Storage, you must use an auth token authentication in the credential object.

Exceptions

Table 6-20 SEND_MESSAGE Function Exception

Exception	Description
Null pipe name	Permission error. Insufficient privilege to write to the pipe. The pipe is private and owned by someone else.

SET_CREDENTIAL_NAME Procedure

This procedure sets the `credential_name` variable that is used as a default credential when pipe messages are stored in Cloud Object Store.

Syntax

```
DBMS_PIPE.SET_CREDENTIAL_NAME (
   credential_name    IN VARCHAR2 );
```

Parameters

Parameter	Description
`credential_name`	The name of the credential to access the Cloud Object Storage. The `credential_name` value can be an OCI resource principal, Azure service principal, Amazon Resource Name (ARN), or a Google service account. See Configure Policies and Roles to Access Resources for more information on resource principal based authentication.

Usage Note

If you use Oracle Cloud Infrastructure Object Storage to store messages, you can use Oracle Cloud Infrastructure Native URIs or Swift URIs. However, the location URI and the credential must match in type as follows:

- If you use a native URI format to access Oracle Cloud Infrastructure Object Storage, you must use Native Oracle Cloud Infrastructure Signing Keys authentication in the credential object.
- If you use Swift URI format to access Oracle Cloud Infrastructure Object Storage, you must use an auth token authentication in the credential object.

Example

```
BEGIN
     DBMS_PIPE.SET_CREDENTIAL_NAME(
       credential_name =>  'my_cred1');
END;
/
```

SET_LOCATION_URI Procedure

This procedure sets the global `location_uri` variable.

Syntax

```
DBMS_PIPE.SET_LOCATION_URI (
   location_uri    IN VARCHAR2 );
```

Parameter

Parameter	Description
location_uri	Object or file URI. The format of the URI depends on the Cloud Object Storage service you are using, for details see DBMS_CLOUD URI Formats.

Usage Note

If you use Oracle Cloud Infrastructure Object Storage to store messages, you can use Oracle Cloud Infrastructure Native URIs or Swift URIs. However, the location URI and the credential must match in type as follows:

- If you use a native URI format to access Oracle Cloud Infrastructure Object Storage, you must use Native Oracle Cloud Infrastructure Signing Keys authentication in the credential object.
- If you use Swift URI format to access Oracle Cloud Infrastructure Object Storage, you must use an auth token authentication in the credential object.

Example

```
BEGIN
  DBMS_PIPE.GET_LOCATION_URI(
      location_uri  => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname1/');
END;
/
```

DBMS_CLOUD_PIPELINE Package

The DBMS_CLOUD_PIPELINE package allows you to create data pipelines for loading and exporting data in the cloud. This package supports continuous incremental data load of files in object store into the database. DBMS_CLOUD_PIPELINE also supports continuous incremental export of table data or query results from the database to object store based on a timestamp column.

- Summary of DBMS_CLOUD_PIPELINE Subprograms
  This table summarizes the subprograms included in the DBMS_CLOUD_PIPELINE package.
- DBMS_CLOUD_PIPELINE Attributes
  Attributes help to control and configure the behavior of a data pipeline.
- DBMS_CLOUD_PIPELINE Views
  The DBMS_CLOUD_PIPELINE package uses the following views.

Summary of DBMS_CLOUD_PIPELINE Subprograms

This table summarizes the subprograms included in the DBMS_CLOUD_PIPELINE package.

> **Note:**
>
> You can use the DBMS_CLOUD_PIPELINE package only with Autonomous Database versions 19.22 or later.

Subprogram	Description
CREATE_PIPELINE Procedure	Creates a new data pipeline.
DROP_PIPELINE Procedure	Drops an existing data pipeline.
RESET_PIPELINE Procedure	Resets the tracking state of a data pipeline. Use reset pipeline to restart the pipeline from the initial state of data load or export. Optionally reset pipeline can purge data in the database or in object store, depending on the type of pipeline.
RUN_PIPELINE_ONCE Procedure	Performs an on-demand run of the pipeline in the current foreground session, instead of a scheduled job.
SET_ATTRIBUTE Procedure	Sets pipeline attributes. There are two overloaded procedures, one to set a single attribute and another to set multiple attributes using a JSON document of attribute name/value pairs
START_PIPELINE Procedure	Starts the data pipeline. When a pipeline is started, the pipeline operation will continuously run in a scheduled job according to the "interval" configured in pipeline attributes.
STOP_PIPELINE Procedure	Stops the data pipeline. When a pipeline is stopped, no future jobs are scheduled for the pipeline.

- CREATE_PIPELINE Procedure
  The procedure creates a new data pipeline.
- DROP_PIPELINE Procedure
  The procedure drops an existing data pipeline. If a pipeline has been started, then it must be stopped before it can be dropped.
- RESET_PIPELINE Procedure
  Resets the tracking state of a data pipeline. Use reset pipeline to restart the pipeline from the initial state of data load or export. Optionally reset pipeline can purge data in the database or in object store, depending on the type of pipeline. A data pipeline must be in stopped state to reset it.
- RUN_PIPELINE_ONCE Procedure
  This procedure performs an on-demand run of the pipeline in the current foreground session, instead of a running in a scheduled job. Use `DBMS_CLOUD_PIPELINE.RUN_PIPELINE_ONCE` to test a pipeline before you start the pipeline as a continuous job.
- SET_ATTRIBUTE Procedure
  This procedure sets pipeline attributes. There are two overloaded procedures, one to set a single attribute and another to set multiple attributes using a JSON document of attribute name/value pairs.
- START_PIPELINE Procedure
- STOP_PIPELINE Procedure
  The procedure stops the data pipeline. When a pipeline is stopped, no future jobs are scheduled for the pipeline.

CREATE_PIPELINE Procedure

The procedure creates a new data pipeline.

Syntax

```
DBMS_CLOUD_PIPELINE.CREATE_PIPELINE(
      pipeline_name        IN   VARCHAR2,
      pipeline_type        IN   VARCHAR2,
      attributes           IN   CLOB        DEFAULT NULL,
      description          IN   VARCHAR2    DEFAULT NULL
);
```

Parameters

Parameter	Description
`pipeline_name`	Specifies a name for the pipeline. The pipeline name must follow the naming rules of Oracle SQL identifiers. See Identifiers for more information. This parameter is mandatory.
`pipeline_type`	Specifies the pipeline type. Valid values: `LOAD`, `EXPORT` This parameter is mandatory.
`attributes`	Pipeline attributes in JSON format. Default value: `NULL` See DBMS_CLOUD_PIPELINE Attributes for more information.
`description`	Description for the pipeline. Default value: `NULL`

DROP_PIPELINE Procedure

The procedure drops an existing data pipeline. If a pipeline has been started, then it must be stopped before it can be dropped.

Syntax

```
DBMS_CLOUD_PIPELINE.DROP_PIPELINE(
       pipeline_name        IN   VARCHAR2,
       force                IN   BOOLEAN DEFAULT FALSE
 );
```

Parameters

Parameter	Description
`pipeline_name`	Specifies a pipeline name. This parameter is mandatory.
`force`	Forcibly drop a pipeline, even if it is in started state. Valid values: `TRUE`, `FALSE` Default value: `FALSE`

Usage Note

- In order to drop a pipeline that is in started state, set the `force` parameter to `TRUE`.

RESET_PIPELINE Procedure

Resets the tracking state of a data pipeline. Use reset pipeline to restart the pipeline from the initial state of data load or export. Optionally reset pipeline can purge data in the database or in object store, depending on the type of pipeline. A data pipeline must be in stopped state to reset it.

Syntax

```
DBMS_CLOUD_PIPELINE.RESET_PIPELINE(
       pipeline_name        IN   VARCHAR2,
       purge_data           IN   BOOLEAN DEFAULT FALSE
 );
```

Parameters

Parameter	Description
`pipeline_name`	Specifies a name for the pipeline. This parameter is mandatory.
`purge_data`	Purge data applies for either a load pipeline or an export pipeline: • For a load pipeline, when `TRUE`, truncate the data in database table. • For an export pipeline, when `TRUE`, delete the files in the object store location. Valid values: `TRUE`, `FALSE` Default value: `FALSE`

Usage Notes

- A data pipeline must be in stopped state to reset it. See STOP_PIPELINE Procedure for more information.
- For a load pipeline, resetting the pipeline clears the record of the files being loaded by the pipeline. When you call either `START_PIPELINE` or `RUN_PIPELINE_ONCE` after resetting a load pipeline, the pipeline repeats the data load and includes all the files present in the object store location.

  When `purge_data` is set to `TRUE`, `DBMS_CLOUD_PIPELINE.RESET_PIPELINE` does the following:

  – Truncates the data in the pipeline's database table you specify with `table_name` attribute.

  – Drops the pipeline's status table, and the pipeline's bad file table and error table (if they exist).
- For an export pipeline, resetting the pipeline clears the last tracked data in the database table. When you call either `START_PIPELINE` or `RUN_PIPELINE_ONCE` after resetting an export pipeline, the pipeline repeats exporting data from the table or query.

  When `purge_data` set to `TRUE`, `DBMS_CLOUD_PIPELINE.RESET_PIPELINE` deletes existing files in the object store location specified with the `location` attribute.

RUN_PIPELINE_ONCE Procedure

This procedure performs an on-demand run of the pipeline in the current foreground session, instead of a running in a scheduled job. Use `DBMS_CLOUD_PIPELINE.RUN_PIPELINE_ONCE` to test a pipeline before you start the pipeline as a continuous job.

Syntax

```
DBMS_CLOUD_PIPELINE.RUN_PIPELINE_ONCE(
       pipeline_name   IN  VARCHAR2
);
```

Parameters

Parameter	Description
`pipeline_name`	Specifies a name for the pipeline to run. This parameter is mandatory.

Usage Notes

- After you perform a test run of a pipeline you can reset the pipeline state using `DBMS_CLOUD_PIPELINE.RESET_PIPELINE`. This allows you to reset the pipeline state before you start the pipeline in a scheduled job.
- If a pipeline is in the started state, then it cannot be run in the foreground session.

SET_ATTRIBUTE Procedure

This procedure sets pipeline attributes. There are two overloaded procedures, one to set a single attribute and another to set multiple attributes using a JSON document of attribute name/value pairs.

Syntax

```
PROCEDURE DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
      pipeline_name         IN   VARCHAR2,
      attribute_name        IN   VARCHAR2,
      attribute_value       IN   CLOB
);

PROCEDURE DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
      pipeline_name         IN   VARCHAR2,
      attributes            IN   CLOB
);
```

Parameters

Parameter	Description
`pipeline_name`	Specifies a name for the pipeline to set attributes. This parameter is mandatory.

Parameter	Description
attribute_name	Specifies the attribute name for the attribute to be set. See DBMS_CLOUD_PIPELINE Attributes for more information.
attribute_value	Specifies the value for the pipeline attribute to set. See DBMS_CLOUD_PIPELINE Attributes for more information.
attributes	Specifies a JSON document containing attribute names and values. See DBMS_CLOUD_PIPELINE Attributes for more information.

Usage Note

- When you use DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE to set multiple attributes with the attributes parameter, all the existing attributes are deleted and overwritten with the specified attributes from the JSON document.

START_PIPELINE Procedure

The procedure starts the data pipeline. When a pipeline is started, the pipeline operation runs continuously in a scheduled job according to the interval configured with the pipeline attributes.

Syntax

```
DBMS_CLOUD_PIPELINE.START_PIPELINE(
      pipeline_name         IN   VARCHAR2,
      start_date            IN   TIMESTAMP WITH TIME ZONE DEFAULT NULL
);
```

Parameters

Parameter	Description
pipeline_name	Specifies a name for the pipeline. This parameter is mandatory.
start_date	Specifies the starting date for the pipeline job. Default value: NULL

Usage Notes

- By default, a pipeline job begins immediately, as soon as the pipeline is started. To start a pipeline job at a later time specify a valid date or timestamp using the start_date parameter.
- See DBMS_CLOUD_PIPELINE Attributes for information on the pipeline interval and other pipeline attributes.

STOP_PIPELINE Procedure

The procedure stops the data pipeline. When a pipeline is stopped, no future jobs are scheduled for the pipeline.

Syntax

```
DBMS_CLOUD_PIPELINE.STOP_PIPELINE(
      pipeline_name        IN   VARCHAR2,
      force                IN   BOOLEAN  DEFAULTFALSE
);
```

Parameters

Parameter	Description
`pipeline_name`	Specifies a name for the pipeline. This parameter is mandatory.
`force`	If force parameter is passed as `TRUE`, then it will terminate any running jobs for the pipeline. Valid values: `TRUE, FALSE` Default value: `FALSE`

DBMS_CLOUD_PIPELINE Attributes

Attributes help to control and configure the behavior of a data pipeline.

Attributes

> **Note:**
>
> As indicated in the **Pipeline Type** column, depending on the pipeline type `LOAD` or `EXPORT`, a pipeline supports a different set of attributes.

Attribute Name	Description	Pipeline Type	Modifiable After Pipeline Starts
`credential_name`	The name of the credential to access the source Cloud Object Storage. You can use `'OCI$RESOURCE_PRINCIPAL'` as the `credential_name` when resource principal is enabled. Credentials specified with the following are also supported: Amazon Resource Names, Azure Service Principal or Google Service Account. See Configure Policies and Roles to Access Resources for more information. Default value: `NULL`. If you do not supply a `credential_name`, `credential_name` is set to `NULL`.	`LOAD`, `EXPORT`	Yes

Attribute Name	Description	Pipeline Type	Modifiable After Pipeline Starts
`field_list`	Identifies the fields in the source files and their data types. This argument's syntax is the same as the `field_list` clause in regular Oracle external tables. For more information about `field_list` see *Oracle® Database Utilities*. Default value: `NULL` The default value specifies the fields and their data types are determined by the columns in the table specified in `table_name` attribute.	`LOAD`	Yes
`format`	The options describing the format for the type of pipeline. • For a load pipeline, see DBMS_CLOUD Package Format Options. • For an export pipeline, see DBMS_CLOUD Package Format Options for EXPORT_DATA Datapump `format` is not supported for an export pipeline. This attribute is mandatory for both `LOAD` and `EXPORT` pipelines.	`LOAD`, `EXPORT`	Yes
`interval`	The time interval in minutes between consecutive executions of scheduled pipeline job. Default value: 15 minutes	`LOAD`, `EXPORT`	Yes
`key_column`	A timestamp or date column in the specified `table` or `query` for exporting newer data continuously to object store. The last execution timestamp or date is tracked by Export pipeline and compared with the value in the `key_column` to identify new data for exporting to object store. Default value: NULL If `key_column` is not specified for an export pipeline, the entire contents of the `table` or `query` are uploaded to object store in each pipeline job execution.	`EXPORT`	No
`location`	Specifies a URI that points to an Object Storage location. The format of the URI depends on the Cloud Object Storage service. See DBMS_CLOUD URI Formats for more information. This attribute is mandatory for both `LOAD` and `EXPORT` pipelines.	`LOAD`, `EXPORT`	No

Attribute Name	Description	Pipeline Type	Modifiable After Pipeline Starts
priority	Specifies a string value that determines the number of parallel operations performed for the pipeline. • In a load pipeline, it determines the number of files loaded in parallel. • In an export pipeline, it determines the degree of parallelism for fetching data from the database. An operation with a higher priority consumes more database resources and is completed sooner. Valid values: • HIGH: Determines the number of parallel files handled using the database's **ECPU count** (**OCPU count** if your database uses OCPUs). • MEDIUM: Determines the number of simultaneous processes using the concurrency limit for Medium service. The default value is 4. • LOW: Run the pipeline job in serial order. Default value: MEDIUM The maximum number of concurrent file operations is limited to 64.	LOAD, EXPORT	Yes
query	Specifies a SELECT statement so that only the required data is exported. The query determines the contents of the files you export as text files (CSV, JSON, Parquet, or XML) or dump files. For example: `SELECT warehouse_id, quantity FROM inventories` Default value: NULL For an export pipeline, either table_name or query is mandatory.	EXPORT	No
table_name	Specifies the name of the target table for loading or exporting data. For a load pipeline table_name is mandatory. For an export pipeline, either table_name or query is mandatory.	LOAD, EXPORT	No
table_owner	The name of the schema where the target table resides for loading or exporting data. Default value: NULL With a NULL value the target table is in the same schema as the user running the procedure.	LOAD, EXPORT	No

DBMS_CLOUD_PIPELINE Views

The DBMS_CLOUD_PIPELINE package uses the following views.

- DBA_CLOUD_PIPELINES View

- USER_CLOUD_PIPELINES View
- DBA_CLOUD_PIPELINE_HISTORY View
- USER_CLOUD_PIPELINE_HISTORY View
- DBA_CLOUD_PIPELINE_ATTRIBUTES View
- USER_CLOUD_PIPELINE_ATTRIBUTES View

DBMS_CLOUD_REPO Package

The `DBMS_CLOUD_REPO` package provides for use of and management of cloud hosted code repositories from Oracle Database. Supported cloud code repositories include GitHub, AWS CodeCommit , and Azure Repos.

- DBMS_CLOUD_REPO Overview
  The `DBMS_CLOUD_REPO` package provides easy access to files in Cloud Code (Git) Repositories, including: GitHub, AWS CodeCommit, and Azure Repos.
- DBMS_CLOUD_REPO Data Structures
  The `DBMS_CLOUD_REPO` package defines record types and a generic JSON object type `repo`.
- DBMS_CLOUD_REPO Subprogram Groups
  The `DBMS_CLOUD_REPO` package subprograms can be grouped into four categories: Initialization Operations, Repository Management Operations, File Operations, and SQL Install Operations.
- Summary of DBMS_CLOUD_REPO Subprograms
  This section covers the `DBMS_CLOUD_REPO` subprograms provided with Autonomous Database.

DBMS_CLOUD_REPO Overview

The `DBMS_CLOUD_REPO` package provides easy access to files in Cloud Code (Git) Repositories, including: GitHub, AWS CodeCommit, and Azure Repos.

This package is a single interface for access to Multicloud Code repositories and allows you to upload SQL files to Git repositories or install SQL scripts directly from Cloud Code Repositories. This package also allows you to use a Cloud Code Repository to manage code versions for SQL scripts and to install or patch application code from Git repositories.

Concepts

- Git Version Control System: Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows.
- Git Repository: A Git repository is a virtual storage of your project. It allows you to save versions of your code, which you can access when needed.

Architecture

`DBMS_CLOUD_REPO` package provides four feature areas:

- Repository Initialization with Generic Cloud Code Repository Handle
  - Initialize a GitHub Code Repository
  - Initialize an AWS CodeCommit Code Repository

  - Initialize an Azure Repos Code Repository
- Repository Management Operations
  - Create a repository
  - Update a repository
  - List repositories
  - Delete a repository
- Repository File Management Operations
  - Upload a file to Code Repository from Oracle Database.
  - Download a file from Code Repository to Oracle Database.
  - Delete files from Code Repository.
  - List files from Code Repository.
- SQL Install Operations
  - Export Database object metadata DDL to repository.
  - Install SQL statements from a file in the Code Repository in Oracle Database.
  - Install SQL statements from a buffer.

DBMS_CLOUD_REPO Data Structures

The `DBMS_CLOUD_REPO` package defines record types and a generic JSON object type `repo`.

REPO JSON Object

A `DBMS_CLOUD_REPO REPO` is an opaque JSON object to represent a Cloud Code Repository of a specific cloud provider. A REPO object can be passed to different `DBMS_CLOUD_REPO` APIs. This opaque object ensures that `DBMS_CLOUD_REPO` procedures and functions are multicloud compatible; you do not have to change any code when you migrate from one Cloud Code Repository provider to another Cloud Code Repository.

DBMS_CLOUD_REPO Subprogram Groups

The `DBMS_CLOUD_REPO` package subprograms can be grouped into four categories: Initialization Operations, Repository Management Operations, File Operations, and SQL Install Operations.

- DBMS_CLOUD_REPO Initialization Operations
  Lists the subprograms for initialization operations within the `DBMS_CLOUD_REPO` package.
- DBMS_CLOUD_REPO Repository Management Operations
  Shows the subprograms for repository management operations within the `DBMS_CLOUD_REPO` package.
- DBMS_CLOUD_REPO Repository Branch Management Operations
  Lists the subprograms for repository branch management operations within the `DBMS_CLOUD_REPO` package.
- DBMS_CLOUD_REPO File Operations
  Lists the subprograms for file operations within the `DBMS_CLOUD_REPO` package.
- DBMS_CLOUD_REPO SQL Install Operations
  Lists the subprograms for SQL install operations within the `DBMS_CLOUD_REPO` package.

DBMS_CLOUD_REPO Initialization Operations

Lists the subprograms for initialization operations within the `DBMS_CLOUD_REPO` package.

Subprogram	Description
INIT_AWS_REPO Function	This function initializes an AWS repository handle and returns an opaque type.
INIT_AZURE_REPO Function	This function initializes an Azure repository handle and returns an opaque type.
INIT_GITHUB_REPO Function	This function initializes a GitHub repository handle and returns an opaque type.
INIT_REPO Function	This function initializes a Cloud Code Repository handle and returns an opaque JSON object.

DBMS_CLOUD_REPO Repository Management Operations

Shows the subprograms for repository management operations within the `DBMS_CLOUD_REPO` package.

Subprogram	Description
CREATE_REPOSITORY Procedure	This procedure creates a Cloud Code Repository identified by the `repo` handle argument.
DELETE_REPOSITORY Procedure	This procedure deletes the Cloud Code Repository identified by the `repo` handle argument.
LIST_REPOSITORIES Function	This function lists all the Cloud Code Repositories identified by the `repo` handle argument.
UPDATE_REPOSITORY Procedure	This procedure updates a Cloud Code repository identified by the `repo` handle argument. The procedure supports updating the name, description, or private visibility status, as supported by the Cloud Code repository.

DBMS_CLOUD_REPO Repository Branch Management Operations

Lists the subprograms for repository branch management operations within the `DBMS_CLOUD_REPO` package.

Subprogram	Description
CREATE_BRANCH Procedure	This procedure creates a branch in a Cloud Code Repository identified by the `repo` handle argument.
DELETE_BRANCH Procedure	This procedure deletes a branch in a Cloud Code Repository identified by the `repo` handle argument.
LIST_BRANCHES Function	This function lists all the Cloud Code Repository branches identified by the `repo` handle argument.
LIST_COMMITS Function	This function lists all the commits in a Cloud Code Repository branch identified by the `repo` handle argument.
MERGE_BRANCH Procedure	This procedure merges a Cloud Code Repository branch into another specified branch in a Cloud Code Repository identified by the `repo` handle argument.

DBMS_CLOUD_REPO File Operations

Lists the subprograms for file operations within the `DBMS_CLOUD_REPO` package.

Subprogram	Description
DELETE_FILE Procedure	This procedure deletes a file from the Cloud Code repository identified by the `repo` handle argument.
GET_FILE Procedure and Function	The function downloads the contents of a file from the Cloud Code repository. The procedure allows you to download the contents of a file from the Cloud Code repository and save the file in a directory.
LIST_FILES Function	This function downloads a file from Cloud Code repository. Optionally, file content can be accessed from either a specific branch, tag or commit name. By default, the file is accessed from the default repository branch.
PUT_FILE Procedure	This procedure uploads a file to the Cloud Code repository identified by the `repo` handle argument. The procedure is overloaded to support either uploading a file from a directory object or uploading the contents from a CLOB to the repository file.

DBMS_CLOUD_REPO SQL Install Operations

Lists the subprograms for SQL install operations within the `DBMS_CLOUD_REPO` package.

Subprogram	Description
EXPORT_OBJECT Procedure	This procedure uploads the DDL metadata of a database object to the Cloud Code repository identified by the `repo` handle argument.
EXPORT_SCHEMA Procedure	This procedure exports metadata of all objects in a schema to a Cloud Code Repository branch identified by the `repo` handle argument.
INSTALL_FILE Procedure	This procedure installs SQL statements from a file in the Cloud Code repository identified by the `repo` handle argument.
INSTALL_SQL Procedure	This procedure installs SQL statements from a buffer given as input.

Summary of DBMS_CLOUD_REPO Subprograms

This section covers the `DBMS_CLOUD_REPO` subprograms provided with Autonomous Database.

The `DBMS_CLOUD_REPO` package is made up of the following:

- DBMS_CLOUD_REPO Initialization Operations
- DBMS_CLOUD_REPO Repository Management Operations
- DBMS_CLOUD_REPO File Operations
- DBMS_CLOUD_REPO SQL Install Operations
- CREATE_BRANCH Procedure
  This procedure creates a branch in the Cloud Code Repository identified by the `repo` handle argument.
- CREATE_REPOSITORY Procedure
  This procedure creates a Cloud Code Repository identified by the `repo` handle argument.
- DELETE_BRANCH Procedure
  This procedure deletes a branch in the Cloud Code repository identified by the `repo` handle argument.
- DELETE_FILE Procedure
  This procedure deletes a file from the Cloud Code repository identified by the `repo` handle argument.

- DELETE_REPOSITORY Procedure
  This procedure deletes the Cloud Code Repository identified by the `repo` handle argument.
- EXPORT_OBJECT Procedure
  This procedure uploads the DDL metadata of a database object to the Cloud Code repository identified by the `repo` handle argument. This procedure is an easy way to upload the metadata definition of a database object in single step.
- EXPORT_SCHEMA Procedure
  This procedure exports metadata of all objects in a schema to the Cloud Code Repository branch identified by the `repo` handle argument.
- GET_FILE Procedure and Function
  The function downloads the contents of a file from the Cloud Code repository. The procedure allows you to download the contents of a file from the Cloud Code repository and save the file in a directory.
- INIT_AWS_REPO Function
  This function initializes an AWS repository handle and returns an opaque type.
- INIT_AZURE_REPO Function
  This function initializes an Azure repository handle and returns an opaque type. This function is only supported for Azure cloud provider.
- INIT_GITHUB_REPO Function
  This function initializes a GitHub repository handle and returns an opaque type.
- INIT_REPO Function
  This function initializes a Cloud Code Repository handle and returns an opaque JSON object. This function is a generic interface to accept a JSON document, and avoids having to change code, you only need to change a JSON document, when moving a code repository from one Cloud Code repository to another Cloud Code repository.
- INSTALL_FILE Procedure
  This procedure installs SQL statements from a file in the Cloud Code repository identified by the `repo` handle argument.
- INSTALL_SQL Procedure
  This procedure installs SQL statements from a buffer given as input.
- LIST_BRANCHES Function
  This function lists branches in the Cloud Code Repository branch identified by the `repo` handle argument.
- LIST_COMMITS Function
  This function lists commits in the Cloud Code Repository branch identified by the `repo` handle argument.
- LIST_FILES Function
  This function downloads a file from Cloud Code repository. Optionally, file content can be accessed from either a specific branch, tag or commit name. By default, the file is accessed from the default repository branch. The results include the file names and additional metadata about the files.
- LIST_REPOSITORIES Function
  This function lists all the Cloud Code Repositories identified by the `repo` handle argument. The results include the repository names and additional metadata about the repositories.
- MERGE_BRANCH Procedure
  This procedure merges a repository branch into another specified branch in the Cloud Code Repository identified by the `repo` handle argument. The `MERGE_BRANCH` procedure is currently not supported in Azure.

- PUT_FILE Procedure
  This procedure uploads a file to the Cloud Code repository identified by the `repo` handle argument. The procedure is overloaded to support either uploading a file from a directory object or uploading the contents from a BLOB to the repository file.
- UPDATE_REPOSITORY Procedure
  This procedure updates a Cloud Code repository identified by the `repo` handle argument. UPDATE_REPOSITORY supports updating the name, description, or private visibility status, as supported by the Cloud Code repository.

CREATE_BRANCH Procedure

This procedure creates a branch in the Cloud Code Repository identified by the `repo` handle argument.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.CREATE_BRANCH(
    repo               IN   CLOB,
    branch_name        IN   VARCHAR2,
    parent_branch_name IN   VARCHAR2 DEFAULT NULL,
    parent_commit_id   IN   VARCHAR2 DEFAULT NULL
 );
```

Parameters

Parameter	Description
`repo`	Specifies the repository handle. This parameter is mandatory and supported for all cloud providers.
`branch_name`	Specifies the repository branch name. This parameter is mandatory and supported for all cloud providers.
`parent_branch_name`	Creates the new branch using the head commit of the specified parent branch. This parameter is supported for all cloud providers. If you do not supply a `parent_branch_name` value, the `parent_branch_name` is set to `main`.
`parent_commit_id`	Creates the new branch using the specified repository commit. This parameter is supported for all cloud providers. If you do not supply a `parent_commit_id` value, the `parent_commit_id` is set to a NULL value.

> **Note:**
>
> To create a branch in a Cloud Code repository, you must specify the parent branch or the parent commit id.

Example

```
BEGIN
  DBMS_CLOUD_REPO.CREATE_BRANCH (
```

```
    repo                => l_repo,
    branch_name         => 'test_branch',
    parent_branch_name  => 'main'
    );
END;
/
```

Usage Note

To run the `DBMS_CLOUD_REPO.CREATE_BRANCH` procedure, you must be logged in as the ADMIN user or have `EXECUTE` privilege on `DBMS_CLOUD_REPO`.

CREATE_REPOSITORY Procedure

This procedure creates a Cloud Code Repository identified by the `repo` handle argument.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.CREATE_REPOSITORY(
      repo                IN   CLOB,
      description         IN   CLOB     DEFAULT NULL,
      private             IN   BOOLEAN  DEFAULT TRUE
);
```

Parameters

Parameter	Description
`repo`	Specifies the repository handle. This parameter is supported for all cloud providers.
`description`	A short text description for the repository. This parameter is supported for GITHUB and AWS cloud provider.
`private`	Repository is private and only accessible with valid credentials This parameter is only supported for the GITHUB cloud provider.

Example

```
BEGIN
  DBMS_CLOUD_REPO.CREATE_REPOSITORY(
    repo        => l_repo,
    description => 'My test repo',
    private => TRUE
    );
END;
/
```

DELETE_BRANCH Procedure

This procedure deletes a branch in the Cloud Code repository identified by the `repo` handle argument.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.DELETE_BRANCH (
      repo              IN  CLOB,
      branch_name       IN  VARCHAR2  DEFAULT NULL
 );
```

Parameters

Parameter	Description
repo	Specifies the repository handle. This parameter is mandatory and supported for all cloud providers.
branch_name	Delete branch from a specific repository. This parameter is mandatory and supported for all cloud providers.

Example

```
BEGIN
  DBMS_CLOUD_REPO.DELETE_BRANCH (
      repo        => l_repo,
      branch_name => 'test_branch'
  );
END;
/
```

Usage Note

To run the DBMS_CLOUD_REPO.DELETE_BRANCH procedure, you must be logged in as the ADMIN user or have EXECUTE privilege on DBMS_CLOUD_REPO.

DELETE_FILE Procedure

This procedure deletes a file from the Cloud Code repository identified by the repo handle argument.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.DELETE_FILE(
      repo              IN  CLOB,
      file_path         IN  VARCHAR2,
      branch_name       IN  VARCHAR2  DEFAULT NULL,
      commit_details    IN  CLOB      DEFAULT NULL
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle.
file_path	File path to delete file in the repository.
branch_name	Delete file from a specific branch.

Parameter	Description
commit_details	Commit Details as a JSON document {"message": "Commit message", "author": {"name": "Committing user name", "email": "Email of committing user" } }

Example

```
BEGIN
  DBMS_CLOUD_REPO.DELETE_FILE(
      repo        => l_repo,
      file_path   => 'scripts/test3.sql',
      branch_name => 'test_branch'
  );
END;
/
```

DELETE_REPOSITORY Procedure

This procedure deletes the Cloud Code Repository identified by the `repo` handle argument.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.DELETE_REPOSITORY(
      repo           IN   CLOB
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle.

Example

```
BEGIN
  DBMS_CLOUD_REPO.DELETE_REPOSITORY(
       repo => l_repo
  );
END;
/
```

EXPORT_OBJECT Procedure

This procedure uploads the DDL metadata of a database object to the Cloud Code repository identified by the `repo` handle argument. This procedure is an easy way to upload the metadata definition of a database object in single step.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.EXPORT_OBJECT(
      repo              IN  CLOB,
      file_path         IN  VARCHAR2,
      object_type       IN  VARCHAR2,
      object_name       IN  VARCHAR2 DEFAULT NULL,
      object_schema     IN  VARCHAR2 DEFAULT NULL,
      branch_name       IN  VARCHAR2 DEFAULT NULL,
      commit_details    IN  CLOB     DEFAULT NULL,
      append            IN  BOOLEAN  DEFAULT FALSE
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle.
file_path	File path to upload object metadata in the repository.
object_type	Object type supported by DBMS_METADATA. See DBMS_METADATA: Object Types table for details.
object_name	Name of the database object to retrieve metadata.
object_schema	Owning schema of the database object.
branch_name	Put file to a specific branch.
commit_details	Commit Details as a JSON document:`{"message": "Commit message", "author": {"name": "Committing user name", "email": "Email of committing user" } }`
append	Append metadata DDL to existing file.

Usage Note

For customized control on the object DDL, you can use `DBMS_METADATA.GET_DDL` along with `DBMS_CLOUD_REPO.PUT_FILE`. In order to get metadata definition of the object, the current user must be privileged to retrieve the object metadata. See DBMS_METADATA for the security requirements of the package.

Example

```
BEGIN
  DBMS_CLOUD_REPO.EXPORT_OBJECT(
     repo        => l_repo,
     object_type => 'PACKAGE',
     object_name => 'MYPACK',
     file_path   => 'mypack.sql'
  );
END;
/
```

EXPORT_SCHEMA Procedure

This procedure exports metadata of all objects in a schema to the Cloud Code Repository branch identified by the `repo` handle argument.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.EXPORT_SCHEMA(
     repo            IN CLOB,
     file_path       IN VARCHAR2,
     schema_name     IN VARCHAR2,
     filter_list     IN CLOB         DEFAULT NULL,
     branch_name     IN VARCHAR2     DEFAULT NULL,
     commit_details  IN CLOB         DEFAULT NULL
);
```

Parameters

Parameter	Description
`repo`	Specifies the repository handle. This parameter is mandatory and supported for all cloud providers.
`file_path`	Specifies the name of the schema file to upload to the repo. This parameter is mandatory and supported for all cloud providers.
`schema_name`	Specifies the name of the schema for which a DDL script is to be uploaded to the Cloud Code Repository branch. This parameter is mandatory and supported for all cloud providers.
`filter_list`	Specifies the CLOB of the JSON array that defines the filter conditions to include or exclude the objects whose metadata needs to be exported. This parameter is supported for all cloud providers. JSON parameters for `filter_list` are: • `match_type`: Specifies the type of filter to be applied to the object types or object names. Valid `match_type` values are: – `in/not_in` – `like/not_like` – `equal/not_equal` • `type`: Specifies the type of object by which to filter. • `name`: Specifies the name of the object by which to filter.
`branch_name`	Specifies the repository branch name. This parameter is supported for all cloud providers. If you do not supply a `branch_name` value, the `branch_name` is set to the default repository branch.

Parameter	Description
commit_details	Commit Details as a JSON document {"message": "Commit message", "author": {"name": "Committing user name", "email": "Email of committing user" } } This parameter is supported for all cloud providers. If you do not supply a commit_details value, the commit_details is set to the default commit message that includes the information about the current database session user and database name performing the commit.

Example

```
BEGIN
  DBMS_CLOUD_REPO.EXPORT_SCHEMA(
    repo          => l_repo,
    schema_name   => 'USER1',
    file_path     => 'myschema_ddl.sql'
    filter_list   =>
        to_clob('[
             { "match_type":"equal",
                "type":"table"
             },
             { "match_type":"not_equal",
                "type":"view"
             },
             { "match_type":"in",
                "type":"table",
                "name": " ''EMPLOYEE_SALARY'',''EMPLOYEE_ADDRESS'' "
             },
             { "match_type":"equal",
                "type":"sequence",
                "name": "EMPLOYEE_RECORD_SEQ"
             },
             { "match_type":"like",
                "type":"table",
                "name": "%OFFICE%"
             }
        ]'
     );
  );
END;
/
```

Usage Note

To run the DBMS_CLOUD_REPO.EXPORT_SCHEMA procedure, you must be logged in as the ADMIN user or have EXECUTE privilege on DBMS_CLOUD_REPO.

GET_FILE Procedure and Function

The function downloads the contents of a file from the Cloud Code repository. The procedure allows you to download the contents of a file from the Cloud Code repository and save the file in a directory.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.GET_FILE(
      repo              IN  CLOB,
      file_path         IN  VARCHAR2,
      branch_name       IN  VARCHAR2  DEFAULT NULL,
      tag_name          IN  VARCHAR2  DEFAULT NULL,
      commit_name       IN  VARCHAR2  DEFAULT NULL
) RETURN CLOB;

PROCEDURE DBMS_CLOUD_REPO.GET_FILE(
      repo              IN  CLOB,
      file_path         IN  VARCHAR2,
      directory_name    IN  VARCHAR2,
      target_file_name  IN  VARCHAR2  DEFAULT NULL,
      branch_name       IN  VARCHAR2  DEFAULT NULL,
      tag_name          IN  VARCHAR2  DEFAULT NULL,
      commit_name       IN  VARCHAR2  DEFAULT NULL
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle.
file_path	File path in the repository.
directory_name	Directory object name to save the file contents.
target_file_name	Target file name to save contents in directory.
branch_name	Get file from a specific branch.
tag_name	Get file from a specific Tag.
commit_name	Get file from a specific commit.

Example

```
BEGIN
  DBMS_CLOUD_REPO.GET_FILE(
     repo             =>  l_repo,
     file_path        => 'test3.sql',
     directory_name   => 'DATA_PUMP_DIR',
     target_file_name => 'test2.sql'
  );
END;
/
```

INIT_AWS_REPO Function

This function initializes an AWS repository handle and returns an opaque type.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.INIT_AWS_REPO(
      credential_name IN  VARCHAR2,
      repo_name       IN  VARCHAR2,
      region          IN  VARCHAR2
)  RETURN repo;
```

Parameters

Parameter	Description
credential_name	Credential object specifying AWS CodeCommit accesskey/secretkey.
repo_name	Specifies the repository name.
region	Specifies the AWS region for the CodeCommit repository.

Example

```
BEGIN
  :repo := DBMS_CLOUD_REPO.INIT_AWS_REPO(
                credential_name => 'AWS_CRED',
                repo_name       => 'my_repo',
                region          => 'us-east-1'
           );
END;
/
```

INIT_AZURE_REPO Function

This function initializes an Azure repository handle and returns an opaque type. This function is only supported for Azure cloud provider.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.INIT_AZURE_REPO(
      credential_name IN  VARCHAR2,
      repo_name       IN  VARCHAR2,
      organization    IN  VARCHAR2,
      project         IN  VARCHAR2
)  RETURN repo;
```

Parameters

Parameter	Description
credential_name	Credential object specifying Azure, with a Username and Personal Access Token (PAT).
repo_name	Specifies the repository name.

Parameter	Description
organization	Specifies the Azure DevOps Organization.
project	Azure Team Project name.

Example

```
BEGIN
  :repo := DBMS_CLOUD_REPO.INIT_AZURE_REPO(
                credential_name => 'AZURE_CRED',
                repo_name       => 'my_repo',
                organization    => 'myorg',
                project         => 'myproj',
           );
END;
/
```

INIT_GITHUB_REPO Function

This function initializes a GitHub repository handle and returns an opaque type.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.INIT_GITHUB_REPO(
      credential_name IN  VARCHAR2  DEFAULT NULL,
      repo_name       IN  VARCHAR2,
      owner           IN  VARCHAR2)
RETURN repo;
```

Parameters

Parameter	Description
credential_name	Credential object specifying GitHub. User Email and Personal Access Token (PAT).
repo_name	Specifies the repository name.
owner	Specifies the repository owner.

Example

```
BEGIN
  :repo := DBMS_CLOUD_REPO.INIT_GITHUB_REPO(
                credential_name => 'GITHUB_CRED',
                repo_name       => 'my_repo',
                owner           => 'foo'
           );
END;
/
```

INIT_REPO Function

This function initializes a Cloud Code Repository handle and returns an opaque JSON object. This function is a generic interface to accept a JSON document, and avoids having to change code, you only need to change a JSON document, when moving a code repository from one Cloud Code repository to another Cloud Code repository.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.INIT_REPO(
     params       IN   CLOB)
RETURN CLOB;
```

Parameters

JSON Parameter	Description
`provider`	Cloud code repository provider from the following: `DBMS_CLOUD_REPO.GITHUB_REPO ('GITHUB')` `DBMS_CLOUD_REPO.AWS_REPO ('AWS')` `DBMS_CLOUD_REPO.AZURE_REPO ('AZURE')`
`repo_name`	Specifies the repository name. `DBMS_CLOUD_REPO.PARAM_REPO_NAME`
`owner`	GitHub Repository Owner. `DBMS_CLOUD_REPO.PARAM_OWNER` This parameter is only applicable for GitHub cloud provider.
`region`	AWS Repository Region `DBMS_CLOUD_REPO_PARAM_REGION` This parameter is only applicable for AWS cloud provider.
`organization`	Azure Organization `DBMS_CLOUD_REPO_PARAM_ORGANIZATION` This parameter is only applicable for Azure cloud provider.
`project`	Azure Team Project `DBMS_CLOUD_REPO_PARAM_PROJECT` This parameter is only applicable for Azure cloud provider

Example

```
BEGIN
  :repo := DBMS_CLOUD_REPO.INIT_REPO(
           params => JSON_OBJECT('credential_name' value 'mycred',
                                 'repo_name'       value 'myrepo',
                                 'repo_owner'      value 'foo')
         );
END;
/
```

INSTALL_FILE Procedure

This procedure installs SQL statements from a file in the Cloud Code repository identified by the `repo` handle argument.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.INSTALL_FILE(
      repo              IN  CLOB,
```

```
      file_path          IN  VARCHAR2,
      branch_name        IN  VARCHAR2  DEFAULT NULL,
      tag_name           IN  VARCHAR2  DEFAULT NULL,
      commit_name        IN  VARCHAR2  DEFAULT NULL,
      stop_on_error      IN  BOOLEAN   DEFAULT TRUE
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle.
file_path	File path in the repository.
branch_name	Branch to install file from a specific branch.
tag_name	Tag to install file from a specific Tag.
commit_name	Commit ID to install file from a specific commit.
stop_on_error	Stop executing the SQL statements on first error.

Usage Notes

- You can install SQL statements containing nested SQL from a Cloud Code repository file using the following:
  - `@`: includes a SQL file with a relative path to the ROOT of the repository.
  - `@@`: includes a SQL file with a path relative to the current file.
- The scripts are intended as schema install scripts and not as generic SQL scripts:
  - Scripts cannot contain SQL*Plus client specific commands.
  - Scripts cannot contain bind variables or parameterized scripts.
  - SQL statements must be terminated with a slash on a new line (/).
  - Scripts can contain DDL, DML PLSQL statements, but direct `SELECT` statements are not supported. Using SELECT within a PL/SQL block is supported.

  Any SQL statement that can be run using `EXECUTE IMMEDIATE` will work if it does not contain bind variables or defines.

Example

```
BEGIN
     DBMS_CLOUD_REPO.INSTALL_FILE(
         repo          => l_repo,
         file_path     => 'test3.sql',
         stop_on_error => FALSE
     );
END;
/
```

INSTALL_SQL Procedure

This procedure installs SQL statements from a buffer given as input.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.INSTALL_SQL(
      content           IN  CLOB,
      stop_on_error     IN  BOOLEAN  DEFAULT TRUE
);
```

Parameters

Parameter	Descriptions
content	Is the CLOB containing the SQL statements to run.
stop_on_error	Stop executing the SQL statements on first error.

Usage Notes

- The scripts are intended as schema install scripts and not as generic SQL scripts:
  - Scripts cannot contain SQL*Plus client specific commands.
  - Scripts cannot contain bind variables or parameterized scripts.
  - SQL statements must be terminated with a slash on a new line (/).
  - Scripts can contain DDL, DML PLSQL statements, but direct `SELECT` statements are not supported. Using SELECT within a PL/SQL block is supported.

  Any SQL statement that can be run using `EXECUTE IMMEDIATE` will work if it does not contain bind variables or defines.

Example

```
BEGIN
  DBMS_CLOUD_REPO.INSTALL_SQL(
      content   => 'create table t1 (x varchar2(30))' || CHR(10) || '/',
      stop_on_error => FALSE
  );
END;
/
```

LIST_BRANCHES Function

This function lists branches in the Cloud Code Repository branch identified by the `repo` handle argument.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.LIST_BRANCHES(
    repo           IN   CLOB
) RETURN list_branch_ret_tab PIPELINED PARALLEL_ENABLE;
```

Parameters

Parameter	Description
repo	Specifies the repository handle. This parameter is mandatory and supported for all cloud providers.

Example

```
SELECT * FROM DBMS_CLOUD_REPO.LIST_BRANCHES (repo => l_repo);
```

Usage Notes

- This is a pipelined table function with return type as `list_branch_ret_tab`.
- `DBMS_CLOUD_REPO.LIST_BRANCHES` returns the column: `name`, which indicates the name of the Cloud Code Repository branch.

LIST_COMMITS Function

This function lists commits in the Cloud Code Repository branch identified by the `repo` handle argument.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.LIST_COMMITS(
      repo               IN  CLOB,
      branch_name        IN  VARCHAR2  DEFAULT NULL,
      file_path          IN  VARCHAR2 DEFAULT NULL,
      commit_id          IN  VARCHAR2  DEFAULT NULL
) RETURN list_commit_ret_tab PIPELINED PARALLEL_ENABLE;
```

Parameters

Parameter	Description
repo	Specifies the repository handle. This parameter is mandatory and supported for all cloud providers.
branch_name	List commits from a specific branch. This parameter is supported for all cloud providers. If you do not supply a `branch_name` value, the `branch_name` is set to `main`.
file_path	List files under the specified subfolder path in the repository. This parameter is only supported for Git and Azure cloud providers. If you do not supply a `file_path` value, the `file_path` is set to a NULL value.
commit_id	List files starting from the specified `sha/id` This parameter is supported for all cloud providers. If you do not supply a `commit_id` value, the `commit_id` is set to a NULL value.

Example

```
SELECT name FROM DBMS_CLOUD_REPO.LIST_COMMITS(repo => l_repo);
```

Example

```
SELECT name FROM DBMS_CLOUD_REPO.LIST_COMMITS (
    repo        => l_repo,
    commit_id   => '66dd2b23b74cd0afabd11af66c6aa9c550540ba6',
    file_path   => 'sub_dir/test11.sql'
);
```

Usage Notes

- This is a pipelined table function with a return type as `list_commit_ret_tab`.
- `DBMS_CLOUD_REPO.LIST_COMMITS` returns the column: `commit_id`.

LIST_FILES Function

This function downloads a file from Cloud Code repository. Optionally, file content can be accessed from either a specific branch, tag or commit name. By default, the file is accessed from the default repository branch. The results include the file names and additional metadata about the files.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.LIST_FILES(
      repo              IN  CLOB,
      path              IN  VARCHAR2  DEFAULT NULL,
      branch_name       IN  VARCHAR2  DEFAULT NULL,
      tag_name          IN  VARCHAR2  DEFAULT NULL,
      commit_id         IN  VARCHAR2  DEFAULT NULL
) RETURN list_file_ret_tab PIPELINED PARALLEL_ENABLE;
```

Parameters

Parameter	Description
`repo`	Specifies the repository handle.
`path`	List files under the specified subfolder path in the repository.
`branch_name`	List files from a specific branch.
`tag_name`	List files from a specific Tag.
`commit_name`	List files from a specific commit.

Usage Notes

- This is a pipelined table function with return type as `list_file_ret_tab`.
- `DBMS_CLOUD_REPO.LIST_FILES` returns the columns: `id`, `name`, `url`, and `bytes`.

Example

```
SELECT name FROM DBMS_CLOUD_REPO.LIST_FILES(repo => l_repo);

NAME
-------------------------
test3.sql
```

LIST_REPOSITORIES Function

This function lists all the Cloud Code Repositories identified by the `repo` handle argument. The results include the repository names and additional metadata about the repositories.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.LIST_REPOSITORIES(
      repo                 IN   CLOB
) RETURN list_repo_ret_tab PIPELINED PARALLEL_ENABLE;
```

Parameters

Parameter	Description
`repo`	Specifies the repository handle. This parameter is supported by all cloud providers.
`description`	A short text description for the repository. This parameter is supported the GITHUB and AWS cloud providers.
`private`	Repository is private and only accessible with valid credentials This parameter is supported for the GITHUB cloud provider.

Usage Notes

- This is a pipelined table function with return type as `list_repo_ret_tab`.
- `DBMS_CLOUD_REPO.LIST_REPOSITORIES` returns the columns: `id`, `name`, `owner`, `description`, `private`, `url`, `bytes`, `created`, and `last_modified`.

Example

```
SELECT name description FROM DBMS_CLOUD_REPO.LIST_REPOSITORIES(:repo);

NAME                 DESCRIPTION
-------------------- ---------------
TestRepo1            My test repo
```

MERGE_BRANCH Procedure

This procedure merges a repository branch into another specified branch in the Cloud Code Repository identified by the `repo` handle argument. The `MERGE_BRANCH` procedure is currently not supported in Azure.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.MERGE_BRANCH (
    repo               IN   CLOB,
    branch_name        IN   VARCHAR2,
    parent_branch_name IN   VARCHAR2 DEFAULT NULL,
    commit_details     IN   CLOB      DEFAULT NULL
 );
```

Parameters

Parameter	Description
repo	Specifies the repository handle. This parameter is mandatory and supported for GITHUB and AWS cloud providers.
branch_name	Specifies the Git branch name to merge. This parameter is mandatory and supported for all cloud providers.
target_branch_name	Specifies the target branch name to merge into. This parameter is mandatory and supported for all cloud providers.
commit_details	Commit Details as a JSON document {"message": "Commit message", "author": {"name": "Committing user name", "email": "Email of committing user" } } If you do not supply a commit_detailsvalue, the commit_details is set to the default commit message that includes the information about current database session user and database name performing the commit.

Example

```
BEGIN
  DBMS_CLOUD_REPO.MERGE_BRANCH (
    repo               => l_repo,
    branch_name        => 'test_branch',
    target_branch_name => 'main'
    );
END;
/
```

Usage Note

To run the DBMS_CLOUD_REPO.MERGE_BRANCH procedure, you must be logged in as the ADMIN user or have EXECUTE privilege on DBMS_CLOUD_REPO.

PUT_FILE Procedure

This procedure uploads a file to the Cloud Code repository identified by the repo handle argument. The procedure is overloaded to support either uploading a file from a directory object or uploading the contents from a BLOB to the repository file.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.PUT_FILE(
      repo              IN  CLOB,
      file_path         IN  VARCHAR2,
      contents          IN  BLOB,
      branch_name       IN  VARCHAR2  DEFAULT NULL,
      commit_details    IN  CLOB      DEFAULT NULL
);

PROCEDURE DBMS_CLOUD_REPO.PUT_FILE(
      repo              IN  CLOB,
      file_path         IN  VARCHAR2,
      directory_name    IN  VARCHAR2,
      source_file_name  IN  VARCHAR2  DEFAULT NULL,
      branch_name       IN  VARCHAR2  DEFAULT NULL,
      commit_details    IN  CLOB      DEFAULT NULL
);
```

Parameters

Parameter	Description
`repo`	Specifies the repository handle.
`file_path`	File path to upload file in the repository.
`contents`	BLOB containing the file contents.
`directory_name`	Directory object name containing the file name.
`source_file_name`	Source file name to upload to repository.
`branch_name`	Put file to a specific branch.
`commit_details`	Commit Details as a JSON document: `{"message": "Commit message", "author": {"name": "Committing user name", "email": "Email of committing user" } }`

Example

```
BEGIN
  DBMS_CLOUD_REPO.PUT_FILE(
     repo   => l_repo,
  );
END;
/
```

UPDATE_REPOSITORY Procedure

This procedure updates a Cloud Code repository identified by the `repo` handle argument. UPDATE_REPOSITORY supports updating the name, description, or private visibility status, as supported by the Cloud Code repository.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.UPDATE_REPOSITORY(
      repo                IN OUT  CLOB,
      new_name            IN      VARCHAR2 DEFAULT NULL,
      description         IN      CLOB     DEFAULT NULL,
      private             IN      BOOLEAN  DEFAULT NULL
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle. This parameter is supported for all cloud providers.
new_name	New name for repository. This parameter is supported for all cloud providers.
description	A short text description for the repository. This parameter is supported for GITHUB and AWS cloud providers.
private	Repository is private and only accessible with valid credentials. This parameter is supported for the GITHUB cloud provider.

Example

```
BEGIN
  DBMS_CLOUD_REPO.UPDATE_REPOSITORY(
     repo        => l_repo,
     new_name    => 'repo2'
  );
END;
/
```

DBMS_DCAT Package

The DBMS_DCAT package provides functions and procedures to help Autonomous Database users leverage the data discovery and centralized metadata management system of OCI Data Catalog.

Data Catalog harvests metadata from a data lake's object storage assets. The harvesting process creates logical entities, which can be thought of as tables with columns and associated data types. DBMS_DCAT procedures and functions connect Autonomous Database to Data Catalog and then synchronize the assets with the database, creating protected schemas and external tables. You can then query object store using those external tables, easily joining external data with data stored in Autonomous Database. This dramatically simplifies the management process; there is a single, centrally managed metadata store that is shared across multiple OCI services (including Autonomous Databases). There are also Autonomous Database dictionary views that allow you to inspect the contents of Data Catalog using SQL, and show you how these Data Catalog entities map to your Autonomous Database schemas and tables.

- Data Catalog Users and Roles
  The DBMS_DCAT package supports synced users/schemas, dcat_admin users and local users. Users must have the dcat_sync role to be able to use this package.

- Required Credentials and IAM Policies
  This topic describes the Oracle Cloud Infrastructure Identity and Access Management (IAM) user credentials and policies required to give Autonomous Database users permission to manage a data catalog and to read from object storage.
- Summary of Connection Management Subprograms
  This table lists the `DBMS_DCAT` package procedures used to create, query and drop Data Catalog connections.
- Summary of Synchronization Subprograms
  Running a synchronization, creating and dropping a synchronization job, and dropping synchronized schemas can be performed with the procedures listed in this table.
- Summary of Data Catalog Views
  Data Catalog integration with Autonomous Database provides numerous tables and views.

Data Catalog Users and Roles

The `DBMS_DCAT` package supports synced users/schemas, `dcat_admin` users and local users. Users must have the `dcat_sync` role to be able to use this package.

Data Catalog Users

- Synced users/schemas
  The synced external tables are organized into database schemas corresponding to Data Asset/Bucket combinations, or according to custom properties set by the user. The synced schemas are automatically created/dropped during Data Catalog synchronization. They are created as no authentication users without the CREATE SESSION privilege. The synced schemas are also created using the protected clause, so that they cannot be altered by local users (not even the PDB admin) and can only be modified through the synchronization.
- User `dcat_admin`
  User `dcat_admin` is a local database user that can run a sync and grant READ privilege on synced tables to other users or roles. The user is created as a no authentication user without the CREATE SESSION privilege.
- Local users
  Database users querying the external tables must be explicitly granted READ privileges on the synced external tables by users `dcat_admin` or ADMIN. By default, after the sync is completed, only users `dcat_admin` and ADMIN have access to the synced external tables.

Data Catalog Roles

- `dcat_sync`
  The `dcat_sync` role has all the required privileges for using the `DBMS_DCAT` package. Users must have this role to be able to use the API for navigating the Data Catalog and running the sync.

Required Credentials and IAM Policies

This topic describes the Oracle Cloud Infrastructure Identity and Access Management (IAM) user credentials and policies required to give Autonomous Database users permission to manage a data catalog and to read from object storage.

OCI Data Catalog Credential and Policy Requirements:

- A credential object with permission to manage a Data Catalog instance is required. Credential objects containing OCI native authentication or resource principals credentials

are supported. Credential objects based on authentication token user principals are not supported.
For information on managing credentials, see DBMS_CLOUD for Access Management.

For OCI native authentication examples, see Example: Creating an OCI Native Authentication Credential Object and Autonomous Database Now Supports Accessing the Object Storage with OCI Native Authentication.

For examples using resource principal, see Example: Using Autonomous Database Resource Principal and Accessing Oracle Cloud Infrastructure Resources from Your Autonomous Database using Resource Principal.

- The manage Data Catalog privilege is required in order for Autonomous Database to add custom properties to the Data Catalog namespace. These privileges allow you to override schema names, table names, column names and more.

  For further information on Data Catalog permissions, see Permissions Required for Each API Operation.

- The read object storage privilege on buckets is required so that Autonomous Database can query data files.

  For further Oracle Object Storage Policy Examples, see Policy Examples.

AWS Glue Data Catalog Credential and Policy Requirements

The following user credentials and policies are required to give Autonomous Database users permission to access Amazon Web Services (AWS) Glue Data Catalogs and to read from the S3 object storage:

- A credential object with permission to access an AWS Glue Data Catalog is required. For information on managing credentials, see DBMS_CLOUD for Access Management.
  For accessing an AWS Glue Data Catalog the following privileges are required: glue:GetDatabases , glue:GetTables , and glue:GetTable.

  In addition, privilege s3:GetBucketLocation is needed during synchronization for generating resolvable https urls pointing to the underlying S3 objects.

- A credential object with permission to access the files stored in S3 is required so that Autonomous Database can query data files.

- AWS credentials are supported. AWS Amazon Resource Names (ARN) credentials are not supported.

Example: Creating an OCI Native Authentication Credential Object

In this example, we create an OCI native authentication credential that can be used when creating a data catalog or an object store credential object. For more details, see DBMS_DCAT SET_DATA_CATALOG_CREDENTIAL Procedure and DBMS_DCAT SET_OBJECT_STORE_CREDENTIAL Procedure respectively.

In OCI native authentication, the `DBMS_CLOUD.CREATE_CREDENTIAL` procedure includes these parameters: `credential_name`, `user_ocid`, `tenancy_ocid`, `private_key`, and `fingerprint`. See DBMS_CLOUD CREATE_CREDENTIAL Procedure for a complete description of this procedure.

The `credential_name` is the name of the credential object. The `user_ocid` and `tenancy_ocid` parameters correspond to the user's and tenancy's OCIDs respectively.

The `private_key` parameter specifies the generated private key in PEM format. Private keys created with a passphrase are not supported. Therefore, we need to make sure we generate a key with no passphrase. See How to Generate an API Signing Key for more details on how to create a private key with no passphrase. Also, the private key that we provide for this

parameter must only contain the key itself without any header or footer (e.g. '-----BEGIN RSA PRIVATE KEY-----', '-----END RSA PRIVATE KEY-----').

The `fingerprint` parameter specifies the fingerprint that is obtained either after uploading the public key to the console, or using the OpenSSL commands. See How to Upload the Public Key and How to Get the Key's Fingerprint for further details on obtaining the fingerprint.

Once all the necessary information is gathered and the private key is generated, we're ready to run the following `CREATE_CREDENTIAL` procedure:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name => 'OCI_NATIVE_CRED',
    user_ocid             =>
'ocid1.user.oc1..aaaaaaaatfn77fe3fxux3o5lego7glqjejrzjsqsrs64f4jsjrhbsk5qzndq'
,
    tenancy_ocid          =>
'ocid1.tenancy.oc1..aaaaaaaapwkfqz3upqklvmelbm3j77nn3y7uqmlsod75rea5zmtmbl574v
e6a',
    private_key           =>
'MIIEogIBAAKCAQEA...t9SH7Zx7a5iV7QZJS5WeFLMUEv+YbYAjnXK+dOnPQtkhOblQwCEY3Hsblj
7Xz7o=',
    fingerprint           =>
'4f:0c:d6:b7:f2:43:3c:08:df:62:e3:b2:27:2e:3c:7a');
END;
/
```

After creating the credential object, it displays in the `dba_credentials` table:

```
SELECT owner, credential_name
FROM dba_credentials
WHERE credential_name LIKE '%NATIVE%';

OWNER CREDENTIAL_NAME
----- ---------------
ADMIN OCI_NATIVE_CRED
```

Example: Using Autonomous Database Resource Principal

In this example, a dynamic group is created that includes appropriate resource members, the dynamic group is given permission to manage a Data Catalog, and then the dynamic group is given permission to read from object storage.

1. Create a dynamic group named `adb-grp-1`. Add a matching rule to `adb-grp-1` that includes the Autonomous Database instance with OCID `ocid1.autonomousdatabase.oc1.iad.abuwcljr...fjkfe` as a resource member.

   Dynamic group matching rule:

   ```
   resource.id = 'ocid1.autonomousdatabase.oc1.iad.abuwcljr...fjkfe'
   ```

2. Define a policy granting the `adb-grp-1` dynamic group full access to the Data Catalog instances, in the `mycompartment` compartment.

```
allow dynamic-group adb-grp-1 to manage data-catalog-family in compartment
mycompartment
```

3. Define a policy that allows the `adb-grp-1` dynamic group to read any bucket in the compartment named `mycompartment`.

```
allow dynamic-group adb-grp-1 to read objects in compartment mycompartment
```

Example: Using User Principals

In this example, `user1` is a member of the group `adb-admins`. All members of this group are given permission to manage all data catalogs in `mycompartment`, and to read from object-store in `mycompartment`.

1. Allow users that are members of `adb-admins` to manage all data catalogs within `mycompartment`.

```
allow group adb-admins to manage data-catalog-family in compartment
mycompartment
```

2. Allow users that are members of `adb-admins` to read any object in any bucket within `mycompartment`.

```
allow group adb-admins to read objects in compartment mycompartment
```

Summary of Connection Management Subprograms

This table lists the `DBMS_DCAT` package procedures used to create, query and drop Data Catalog connections.

Subprogram	Description
SET_DATA_CATALOG_CONN Procedure	Create a connection to the given data catalog
SET_DATA_CATALOG_CREDENTIAL Procedure	Set the data catalog access credential used by a specific connection to the data catalog
SET_OBJECT_STORE_CREDENTIAL Procedure	Set the credential used by the given unique connection identifier for accessing the Object Store
UNSET_DATA_CATALOG_CONN Procedure	Remove an existing Data Catalog connection

- SET_DATA_CATALOG_CREDENTIAL Procedure
  This procedure sets the Data Catalog access credential used by a specific connection to the Data Catalog.
- SET_OBJECT_STORE_CREDENTIAL Procedure
  This procedure sets the credential that is used by the given unique connection identifier for accessing the Object Store. Changing the Object Store access credential alters all existing synced tables to use the new credential.

- SET_DATA_CATALOG_CONN Procedure
  This procedure creates a connection to the given Data Catalog. The connection is required to synchronize metadata with Data Catalog. An Autonomous Database instance can connect to multiple Data Catalog instances and supports connecting to OCI Data Catalogs and AWS Glue Data Catalogs.
- UNSET_DATA_CATALOG_CONN Procedure
  This procedure removes an existing Data Catalog connection.

SET_DATA_CATALOG_CREDENTIAL Procedure

This procedure sets the Data Catalog access credential used by a specific connection to the Data Catalog.

Syntax

```
PROCEDURE DBMS_DCAT.SET_DATA_CATALOG_CREDENTIAL(
    credential_name VARCHAR2(128) DEFAULT NULL,
    dcat_con_id     VARCHAR2 DEFAULT NULL
  );
```

Parameters

Parameter	Description
credential_name	(Optional) The credential used for accessing the Data Catalog.
dcat_con_id	The unique Data Catalog connection identifier. This credential is used for the connection identified by dcat_con_id. The default is Null.

Usage

This credential must have Manage Data Catalog permissions; see Data Catalog Policies. The default is the resource principal; see Access Cloud Resources by Configuring Policies and Roles.

SET_OBJECT_STORE_CREDENTIAL Procedure

This procedure sets the credential that is used by the given unique connection identifier for accessing the Object Store. Changing the Object Store access credential alters all existing synced tables to use the new credential.

Syntax

```
PROCEDURE DBMS_DCAT.SET_OBJECT_STORE_CREDENTIAL(
    credential_name  VARCHAR2(128),
    dcat_con_id      IN VARCHAR2 DEFAULT NULL
  );
```

Parameters

Parameter	Description
credential_name	The credential used by the external tables for accessing the Object Store.
dcat_con_id	The unique Data Catalog connection identifier. The default is NULL.

SET_DATA_CATALOG_CONN Procedure

This procedure creates a connection to the given Data Catalog. The connection is required to synchronize metadata with Data Catalog. An Autonomous Database instance can connect to multiple Data Catalog instances and supports connecting to OCI Data Catalogs and AWS Glue Data Catalogs.

Syntax

```
PROCEDURE DBMS_DCAT.SET_DATA_CATALOG_CONN (
   region       VARCHAR2 DEFAULT NULL,
   endpoint     VARCHAR2 DEFAULT NULL,
   catalog_id   VARCHAR2 DEFAULT NULL,
   dcat_con_id  VARCHAR2 DEFAULT NULL,
   catalog_type VARCHAR2 DEFAULT NULL
 );
```

Parameters

Parameter	Description
`region`	The Data Catalog region. If the `endpoint` is specified, `region` is optional. If both `endpoint` and `region` are given, then the `endpoint` takes precedence. Default is NULL.
`endpoint`	The Data Catalog endpoint. If the `region` is specified, `endpoint` is optional. If both `endpoint` and `region` are given, then the `endpoint` takes precedence. Default is NULL.
`catalog_id`	The unique Oracle Cloud Identifier (OCID) for the Data Catalog instance. When connecting to AWS Glue Data Catalogs, `catalog_id` is optional.
`dcat_con_id`	A unique Data Catalog connection identifier. This identifier is required when connecting to multiple Data Catalogs and is optional when connecting to only one. It is used to refer to the Data Catalog connection in subsequent calls or when querying views. If no identifier is specified this procedure generates a NULL connection identifier. The following restrictions apply for `dcat_con_id`: • It must be unique within the Autonomous Database instance. • It must start with a letter. • It may contain alphanumeric characters, underscores (_), dollar signs ($), and pound signs (#). • It must be at least 16 characters long.
`catalog_type`	The type of data catalog to connect. Allowed values: • `OCI_DCAT` - OCI Data Catalog • `AWS_GLUE` - AWS Glue Data Catalog • `NULL` - The catalog type is automatically detected from the provided region or endpoint.

Usage

You only need to call this procedure once to set the connection. As part of the connection process, Autonomous Database adds custom properties to Data Catalog. These custom properties are accessible to Data Catalog users and allow you to override default names (for schemas, tables and columns) and column data types.

Before creating a connection, credentials must be created and set. For a description of the connection process, see Typical Workflow with Data Catalog for OCI Data Catalogs and User Workflow for Querying with AWS Glue Data Catalog for AWS Glue Data Catalogs.

Example: Connecting with a known OCID

In this example, Autonomous Database is connecting to Data Catalog in the `uk-london-1` region. The `catalog_id` parameter uses the Oracle Cloud Identifier (`ocid`) for the Data Catalog instance. The type of Data Catalog is automatically determined: AWS Glue Data Catalog or OCI Data Catalog.

```
BEGIN
  DBMS_DCAT.SET_DATA_CATALOG_CONN(
    region=>'uk-london-1',
    catalog_id=>'ocid1.datacatalog.oc1.uk-london-1...');
END;
/
```

Example: Connecting to an AWS Glue Data Catalog

A connection is the association between an Autonomous Database instance and an AWS Glue Data Catalog. After a successful connection, the Autonomous Database instance is able to synchronize with AWS Glue. Each AWS account has one AWS Glue Data Catalog per region and each catalog can be accessed using the corresponding service endpoint for each region. An Autonomous Database instance can be associated with an AWS Glue Data Catalog by invoking the API `DBMS_DCAT.SET_DATA_CATALOG_CONN` and specify the endpoint for the region where the catalog resides.

See AWS Glue endpoints and quotas.

In this example, Autonomous Database is connecting to an AWS Glue Data Catalog in the `uk-london-1` region. Because this is an AWS Glue Data Catalog connection, the `catalog_id` parameter is not needed.

```
BEGIN
  DBMS_DCAT.SET_DATA_CATALOG_CONN(
    region=>'uk-london-1',
    catalog_type=>'AWS_GLUE'
END;
/
```

UNSET_DATA_CATALOG_CONN Procedure

This procedure removes an existing Data Catalog connection.

> **Note:**
>
> Invoking this procedure drops all of the protected schemas and external tables that were created as part of previous synchronizations. It does not impact the metadata in Data Catalog.

Syntax

```
PROCEDURE DBMS_DCAT.UNSET_DATA_CATALOG_CONN (
    dcat_con_id IN VARCHAR2 DEFAULT NULL
);
```

Parameters

Parameter	Description
dcat_con_id	The unique Data Catalog connection identifier. Default is Null.

Summary of Synchronization Subprograms

Running a synchronization, creating and dropping a synchronization job, and dropping synchronized schemas can be performed with the procedures listed in this table.

> **Note:**
>
> On April 4, 2022, the `sync_option` and `grant_read` parameters were added to the `DBMS_DCAT.RUN_SYNC` procedure. To ensure correct performance of scheduled sync jobs created prior to that date, you need to drop and recreate the scheduled sync jobs. See DBMS_DCAT.DROP_SYNC_JOB Procedure and DBMS_DCAT.CREATE_SYNC_JOB Procedure.

Subprogram	Description
CREATE_SYNC_JOB Procedure	Create a scheduler job to invoke `RUN_SYNC` periodically
DROP_SYNC_JOB Procedure	Drop an existing sync job for the given unique connection identifier
DROP_SYNCED_SCHEMAS Procedure	Drop all previously synchronized schemas for the given unique connection identifier
RUN_SYNC Procedure	Run a synchronization operation

- RUN_SYNC Procedure
  This procedure runs a synchronization operation and is the entry point to the synchronization. As an input, it takes lists of selected data catalog assets, folders and entities and materializes them by creating, dropping, and altering external tables.
- CREATE_SYNC_JOB Procedure
  This procedure creates a scheduler job to invoke `RUN_SYNC` periodically.
- DROP_SYNC_JOB Procedure
  This procedure drops an existing sync job for the given unique connection identifier.
- DROP_SYNCED_SCHEMAS Procedure
  This procedure drops all previously synchronized schemas for the given unique connection identifier.

RUN_SYNC Procedure

This procedure runs a synchronization operation and is the entry point to the synchronization. As an input, it takes lists of selected data catalog assets, folders and entities and materializes them by creating, dropping, and altering external tables.

The `sync_option` parameter specifies which operation the `RUN_SYNC` procedure performs: `SYNC`, `DELETE` or `REPLACE`. The operation is performed over entities within the scope of the `synced_objects` parameter.

Every call to the `RUN_SYNC` procedure returns a unique `operation_id` that can be used to query the `USER_LOAD_OPERATIONS` view to obtain information about the status of the sync and the corresponding `log_table`. The `DBMS_DCAT$SYNC_LOG` view can be queried for easy access to the `log_table` for the last sync operation executed by the current user. For further details, see DBMS_DCAT$SYNC_LOG View, and Monitoring and Troubleshooting Loads.

> **Note:**
>
> On April 4, 2022, the `sync_option` and `grant_read` parameters were added to the `RUN_SYNC` procedure. To ensure correct performance of scheduled sync jobs created prior to that date, you need to drop and recreate the scheduled sync jobs. See DBMS_DCAT.DROP_SYNC_JOB Procedure and DBMS_DCAT.CREATE_SYNC_JOB Procedure.

Synchronizing Partitioned Logical Entities or Glue Tables

The `RUN_SYNC` procedure creates a partitioned external table for each logical entity or Glue table when all three of the following apply:

1. The OCI data catalog logical entity or Glue table has one or more partitioned attributes.
2. For OCI data catalogs, the logical entity is derived from a prefix-based filename pattern. Partitioned logical entities derived from regex-based patterns are not supported.
3. For OCI data catalogs, the logical entity is based on partitioned data that follows the hive-style or non-hive folder format. Logical entities based on partitioned data that follow the non-hive style format using object names are not supported.
   - Example 1. Logical entities based on harvested objects that follow the Hive style partitioning format with prefix-based filename patterns.

     Consider the following objects:

     ```
     Bucket: MYBUCKET
     cluster1/db1.db/sales/country=USA/year=2020/month=01/sales1.csv
     cluster1/db1.db/sales/country=USA/year=2020/month=01/sales2.csv
     cluster1/db1.db/sales/country=USA/year=2020/month=02/sales1.csv
     ```

     Harvesting the bucket using a filename pattern with a starting folder prefix of `cluster1/db1.db` generates a logical entity named `SALES` with three partition attributes: `country`, `year`, and `month`. The type for partitioned attributes is `Partition` while the type for non-partitioned attributes is `Primitive`.
   - Example 2. Logical entities based on harvested objects that follow the non-Hive style partitioning format with prefix-based filename patterns.

Consider the following objects:

```
Bucket: MYBUCKET
cluster2/db2.db/sales/USA/2020/01/sales1.csv
cluster2/db2.db/sales/USA/2020/01/sales2.csv
cluster2/db2.db/sales/USA/2020/02/sales1.csv
```

Harvesting the bucket using a filename pattern with a starting folder prefix of `cluster2/db2.db` generates a logical entity named `SALES` with three partition attributes: `name0`, name1, and `name2`. The only difference between the generated logical entity compared to Example 1, is that the names of partitioned attributes are auto generated, while in Example 1 they are extracted from the URL (`country`, `year`, and `month` respectively).

For a complete end to end example of synchronizing partitioned logical entities, see Example: A Partitioned Data Scenario.

Incremental Synchronization of Partitioned Logical Entities/Glue Tables

Every call to the `RUN_SYNC` procedure specifies a set of OCI data catalog logical entities or AWS Glue tables to be synced with the database. When a logical entity or Glue table is present in two `RUN_SYNC` calls, the second call preserves and possibly alters existing external tables. The following table shows which logical entity or Glue table changes are supported when the logical entity or Glue table is partitioned:

Logical Entity or Glue Table Change	Action
Addition, removal, or update of a partition	All partitions of the external partitioned table are updated, regardless of whether a change has been detected by the data catalog.
Addition of a partitioned attribute	Adding a partitioned column to an external partitioned table is not supported. An exception is raised.
Deletion of a partition attribute	Dropping a partitioned column from an external partitioned table is not supported. An exception is raised.
Renaming of a partitioned attribute	Renaming a partitioned column in an external partitioned table is not supported. An exception is raised.

Syntax

```
PROCEDURE DBMS_DCAT.RUN_SYNC (
    synced_objects   IN  CLOB,
    sync_option      IN VARCHAR2 DEFAULT 'SYNC',
    error_semantics  IN VARCHAR2 DEFAULT 'SKIP_ERRORS',
    log_level        IN VARCHAR2 DEFAULT 'INFO',
    grant_read       IN VARCHAR2 DEFAULT NULL,
    dcat_con_id      IN VARCHAR2 DEFAULT NULL
 );
PROCEDURE DBMS_DCAT.RUN_SYNC (
    synced_objects   IN  CLOB,
    sync_option      IN VARCHAR2 DEFAULT 'SYNC',
    error_semantics  IN VARCHAR2 DEFAULT 'SKIP_ERRORS',
    log_level        IN VARCHAR2 DEFAULT 'INFO',
    grant_read       IN VARCHAR2 DEFAULT NULL,
    operation_id     OUT NOCOPY NUMBER,
```

```
   dcat_con_id      IN VARCHAR2 DEFAULT NULL
);
```

Parameters

Parameter	Description
synced_objects	This parameter is a JSON document that specifies the data catalog objects to synchronize. For OCI Data Catalogs, the JSON document specifies a set of entities in multiple granularity: data assets, folders (Object Store buckets) or logical entities. It contains an asset_list that is either an array of asset objects or an array containing a single "*" string that stands for 'sync all (object store) data assets in the catalog'. For AWS Glue Data Catalogs, the JSON document specifies a list of tables in multiple granularity: databases, tables. The document specifies a list of databases. Users can restrict the set of tables to be synced by specifying individual tables within a database.
sync_option	(Optional) There are three options: • SYNC (Default) - This option ensures that what is in the data catalog, over the synced_objects scope, is represented in the Autonomous Database. If a logical entity or Glue table was deleted from the data catalog, since the last sync operation, then it is deleted in the Autonomous Database. The following operations are performed over the synced_objects scope: – Adds tables for new data catalog entities – Removes tables for deleted data catalog entities – Updates properties (such as name, columns and data types) for existing tables • DELETE - Deletes tables within the synced_objects scope. • REPLACE - Replaces all currently synced objects with the objects within the synced_objects scope.
error_semantics	(Optional) This parameter specifies the error behavior. If set to SKIP_ERRORS, the sync attempts to continue despite errors encountered for individual entities. If set to STOP_ON_ERROR, the procedure fails on the first encountered error. The default is SKIP_ERRORS.
log_level	(Optional) This parameter specifies the following values in increasing level of logging detail: (OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE, ALL). The default is INFO.
grant_read	(Optional) This parameter is a list of users/roles that are automatically granted READ privileges on all external tables processed by this invocation of RUN_SYNC. All users/roles in the grant_read list are given READ privileges on all new or already existing external tables that correspond to entities specified by the synced_objects parameter. The RUN_SYNC procedure preserves already granted privileges on synced external tables.
operation_id	(Optional) This parameter is used to find the corresponding entry in USER_LOAD_OPERATIONS for the sync and determine the name of the log table. Note: A version of RUN_SYNC that does not return an operation_id is available so users can query USER_LOAD_OPERATIONS for the latest sync.

Parameter	Description
dcat_con_id	This parameter is the unique data catalog connection identifier that was specified when the connection to the data catalog was created. See DBMS_DCAT SET_DATA_CATALOG_CONN Procedure. This parameter identifies which connection is used for synchronization and becomes a part of the derived schema name. See Synchronization Mapping for a description of how the schema name is derived. The parameter default is NULL.

Example: Synchronize All OCI Data Catalog Entities

In the following example, all Data Catalog entities are synchronized.

```
EXEC DBMS_DCAT.RUN_SYNC(synced_objects=>'{"asset_list":["*"]}');
```

Example: synced_objects Parameter for Synchronizing All OCI Data Catalog Data Assets

The following is an example synced_objects parameter for synchronizing all (Object Storage) data assets in the Data Catalog.

```
{"asset_list" : ["*"]}
```

Example: synced_objects Parameter for Synchronizing Specific OCI Data Catalog Data Assets

The following is an example synced_objects parameter for synchronizing two data assets.

```
{"asset_list": [
        {
            "asset_id":"0b320de9-8411-4448-91fb-9e2e7f78fd5f"
        },
        {
            "asset_id":"0b320de9-8411-4448-91fb-9e2e7f74523"
        }
    ]}
```

Example: synced_objects Parameter for Synchronizing Specific OCI Data Catalog Entities within a Data Asset

The following shows an example synced_objects parameter for synchronizing two entities within the data asset.

```
{"asset_list": [
        {
            "asset_id":"0b320de9-8411-4448-91fb-9e2e7f78fd5f",
            "folder_list":[
                "f1",
                "f2"
           ]
        }
    ]}
```

Example: `synced_objects` Parameter for Synchronizing Specific OCI Data Catalog Folders and Entities within a Data Asset

The following shows an example `synced_objects` parameter for synchronizing two folders and two entities within the data asset.

```
{"asset_list":[
        {
            "asset_id":"0b320de9-8411-4448-91fb-9e2e7f78fd5f",
            "entity_list": [
                "entity1",
                "entity2"
            ],
             "folder_list": [
                "f1",
                "f2"
            ]
        }
    ]}
```

Example: `synced_objects` Parameter for Synchronizing All AWS Glue Data Catalog Databases

The following shows an example `synced_objects` parameter for synchronizing all databases in the AWS Glue Data Catalog.

```
{"database_list":["*"]}
```

Example: `synced_objects` Parameter for Synchronizing Two AWS Glue Data Catalog Databases

The following shows an example `synced_objects` parameter for synchronizing two AWS Glue Data Catalog databases.

```
{"database_list":[
    {"database":"tpcdscsv"},
    {"database":"tpcdsparquet"} ]}
```

Example: `synced_objects` Parameter for Synchronizing Three AWS Glue Data Catalog Databases

The following shows an example `synced_objects` parameter for synchronizing three tables from an AWS Glue Data Catalog database.

```
{"database_list":[
 {"database":"tpcdsparquet",
     "table_list": [ "tpcdsparquet_customer",
                     "tpcdsparquet_item",
                     "tpcdsparquet_web_sales" ] } ]}
```

CREATE_SYNC_JOB Procedure

This procedure creates a scheduler job to invoke `RUN_SYNC` periodically.

It takes as input the set of objects to be synced, the error semantics, the log level, and a repeat interval. See DBMS_DCAT RUN_SYNC Procedure for further details on how synchronization works.

There can only be a single sync job. The `CREATE_SYNC_JOB` procedure fails if another job is already specified, unless the force parameter is set to `TRUE`. If force is set to `TRUE` the previous job is dropped.

If a scheduler job attempts to run while another sync is in progress, the scheduler job fails.

> **Note:**
>
> On April 4, 2022, the `sync_option` and `grant_read` parameters were added to the `RUN_SYNC` procedure. To ensure correct performance of scheduled sync jobs created prior to that date, you need to drop and recreate the scheduled sync jobs. See DBMS_DCAT.DROP_SYNC_JOB Procedure and DBMS_DCAT.CREATE_SYNC_JOB Procedure.

Syntax

```
PROCEDURE DBMS_DCAT.CREATE_SYNC_JOB (
    synced_objects   IN CLOB,
    error_semantics  IN VARCHAR2 DEFAULT 'SKIP_ERRORS',
    log_level        IN VARCHAR2 DEFAULT 'INFO',
    repeat_interval  IN VARCHAR2,
    force            IN VARCHAR2 DEFAULT 'FALSE',
    grant_read       IN VARCHAR2 DEFAULT NULL,
    sync_option      IN VARCHAR2 DEFAULT 'SYNC',
    dcat_con_id      IN VARCHAR2 DEFAULT NULL
);
```

Parameters

Parameter	Description
`synced_objects`	A JSON object specifying the objects to be synced, as described in the `RUN_SYNC` procedure.
`error_semantics`	(Optional) Error behavior, as specified for `RUN_SYNC`. Default is `SKIP_ERRORS`.
`log_level`	(Optional) Logging level, as specified for `RUN_SYNC`. Default is `INFO`.
`repeat_interval`	Repeat interval for the job, with the same semantics as the repeat interval parameter of the `DBMS_SCHEDULER.CREATE_JOB` procedure. For details on the `repeat_interval`, see Overview of Creating Jobs.
`force`	(Optional) If `TRUE`, existing sync jobs are deleted first. If `FALSE`, the `CREATE_SYNC_JOB` procedure fails if a sync job already exists. Default is `FALSE`.

Parameter	Description
grant_read	(Optional) List of users/roles to be granted READ on the synced external tables, as described for procedure RUN_SYNC. See DBMS_DCAT.RUN_SYNC Procedure.
sync_option	(Optional) Behavior with respect to entities that have already been synced through a previous RUN_SYNC operation, as described for procedure RUN_SYNC. See DBMS_DCAT.RUN_SYNC Procedure.
dcat_con_id	This parameter is the unique Data Catalog connection identifier that was specified when the connection to Data Catalog was created. See DBMS_DCAT SET_DATA_CATALOG_CONN Procedure. This parameter identifies which connection is used for synchronization and becomes a part of the derived schema name. See Synchronization Mapping for a description of how the schema name is derived. The parameter default is NULL.

DROP_SYNC_JOB Procedure

This procedure drops an existing sync job for the given unique connection identifier.

Syntax

```
PROCEDURE DBMS_DCAT.DROP_SYNC_JOB (
    dcat_con_id IN VARCHAR2 DEFAULT NULL
  );
```

Parameters

Parameter	Description
dcat_con_id	The unique Data Catalog connection identifier. The default is NULL.

DROP_SYNCED_SCHEMAS Procedure

This procedure drops all previously synchronized schemas for the given unique connection identifier.

Syntax

```
PROCEDURE DBMS_DCAT.DROP_SYNCED_SCHEMAS (
    dcat_con_id IN VARCHAR2 DEFAULT NULL
  );
```

Parameters

Parameter	Description
dcat_con_id	The unique Data Catalog connection identifier. The default is NULL.

Summary of Data Catalog Views

Data Catalog integration with Autonomous Database provides numerous tables and views.

These tables and views help you understand:

- Available Data Catalog assets. Get information about any type of Data Catalog asset - including databases, object stores, and more.
- Information about the Data Catalog Object Storage assets and entities that have been synchronized with Autonomous Database. This includes details about how Data Catalog items (assets, folders and entities) map to Autonomous Database objects (i.e. schemas and external tables).
- Metadata sync executions. Review details about sync jobs, including any issues that may have occurred during synchronization.

This table lists the tables and views provided by the `DBMS_DCAT` package.

View	Description
ALL_CLOUD_CATALOG_DATABASES View	Display information about OCI Data Catalog data assets and AWS Glue Data Catalog databases
ALL_CLOUD_CATALOG_TABLES View	Used to display information about data entities for OCI Data Catalogs and tables for AWS Glue Data Catalogs
ALL_DCAT_ASSETS View	List data catalog assets that this database is authorized to access
ALL_DCAT_ATTRIBUTES View	List data catalog attributes this database is authorized to access
ALL_DCAT_CONNECTIONS View	A view that contains information about the data catalog(s) connected to this instance
ALL_DCAT_ENTITIES View	Lists logical entities this database is authorized to access
ALL_DCAT_FOLDERS View	List metadata for the Object Storage buckets containing the data files for the Logical Entities
ALL_DCAT_GLOBAL_ACCESSIBLE_CATALOGS View	List all accessible catalogs across all regions, along with the level of access privileges for each catalog
ALL_DCAT_LOCAL_ACCESSIBLE_CATALOGS View	List all accessible catalogs in the current region, along with the level of access privileges for each catalog
ALL_GLUE_DATABASES View	Lists the AWS Glue Data Catalog databases that the data catalog credential is authorized to access
ALL_GLUE_TABLES View	Shows all AWS Glue Data Catalog tables that the data catalog credential is authorized to access
DCAT_ATTRIBUTES View	List the mapping of logical entity attributes to external table columns
DCAT_ENTITIES View	Describes the mapping of logical entities to external tables
DBMS_DCAT$SYNC_LOG View	Provides easy access to the log table for the last sync operation executed by the current user

- ALL_CLOUD_CATALOG_DATABASES View
  Use view `ALL_CLOUD_CATALOG_DATABASES` to display information about OCI Data Catalog data assets and AWS Glue Data Catalog databases.
- ALL_CLOUD_CATALOG_TABLES View
  View `ALL_CLOUD_CATALOG_TABLES` is used to display information about data entities for OCI Data Catalogs and tables for AWS Glue Data Catalogs.
- ALL_DCAT_ASSETS View
  The Data Catalog assets that this database is authorized to access.
- ALL_DCAT_ATTRIBUTES View
  The Data Catalog attributes this database is authorized to access.
- ALL_DCAT_CONNECTIONS View
  A view that contains information about the data catalog(s) connected to this instance.
- ALL_DCAT_ENTITIES View
  The Data Catalog logical entities this database is authorized to access.

- ALL_DCAT_FOLDERS View
  Metadata for the Object Storage buckets containing the data files for the Logical Entities.
- ALL_DCAT_GLOBAL_ACCESSIBLE_CATALOGS View
  This view lists all accessible catalogs across all regions, along with the level of access privileges for each catalog.
- ALL_DCAT_LOCAL_ACCESSIBLE_CATALOGS View
  This view lists all accessible catalogs in the current region, along with the level of access privileges for each catalog.
- ALL_GLUE_DATABASES View
  The AWS Glue Data Catalog databases that the data catalog credential is authorized to access.
- ALL_GLUE_TABLES View
  This view shows all AWS Glue Data Catalog tables that the data catalog credential is authorized to access.
- DCAT_ATTRIBUTES View
  Lists the mapping of logical entity attributes to external table columns.
- DCAT_ENTITIES View
  Describes the mapping of logical entities to external tables.
- DBMS_DCAT$SYNC_LOG View
  The `DBMS_DCAT$SYNC_LOG` view provides easy access to the log table for the last sync operation executed by the current user.

ALL_CLOUD_CATALOG_DATABASES View

Use view `ALL_CLOUD_CATALOG_DATABASES` to display information about OCI Data Catalog data assets and AWS Glue Data Catalog databases.

Column	Description
`DCAT_CON_ID`	`CON1`
`CATALOG_ID`	Data catalog unique identifier. **OCI Data Catalog example**: `ocid1.datacatalog.oc1.ap-mumbai-1.….y35a` **AWS Glue Data Catalog example**: `NULL` `579294766787`
`NAME`	Name of the data asset (OCI)/ database (AWS Glue). **OCI Data Catalog example**: `OBJECT_STORE_AT_ASHBURN` **AWS Glue Data Catalog example**: `OBJECT_STORE_AT_N_CALIFORNIA`

Column	Description
DESCRIPTION	Description of the data asset (OCI)/ database (AWS Glue). **OCI Data Catalog example**: `Data stored in S3 (N. California)` **AWS Glue Data Catalog example**: `Data stored in S3 (N. California)`
TIME_CREATED	The date and time the data asset (OCI) / databases (AWS Glue) were created in the data catalog. **OCI Data Catalog example**: `26-SEP-22 10.56.01.395000 PM +00:00` **AWS Glue Data Catalog example**: `2022-06-15T09:45:35+01:00`

Column	Description
DETAILS	JSON document with metadata about each data entity (OCI) / database (AWS Glue). **OCI Data Catalog example**: `{` `  "catalog-id": "ocid1.datacatalog.oc1.ap-mumbai-1.amaaa...",` `  "description": null,` `  "display-name": "OBJECT_STORE_AT_ASHBURN",` `  "external-key": "https://swiftobjectstorage.us-ashburn-1....",` `  "key": "bc95181c-3ac3-4959-9e5f-4e460d3fb82a",` `  "lifecycle-state": "ACTIVE",` `  "time-created": "2022-09-26T22:56:01.395000+00:00",` `  "type-key": "3ea65bc5-f60d-477a-a591-f063665339f9",` `  "uri": "/dcat/20190325/dataAssets/bc95181c-3ac3-4959-9e5f-4e460d3fb82a"` `}` **AWS Glue Data Catalog example**: `{` `    "Name": "dbmsdcatpoc",` `    "Parameters": {` `        "somekey": "somevalue"` `    },` `    "CreateTime": "2022-06-15T09:45:35+01:00",` `    "CreateTableDefaultPermissions": [` `        {` `            "Principal": {` `                "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"` `            },` `            "Permissions": [` `                "ALL"` `            ]` `        }` `    ],` `    "CatalogId": "579294766787"` `}`

ALL_CLOUD_CATALOG_TABLES View

View `ALL_CLOUD_CATALOG_TABLES` is used to display information about data entities for OCI Data Catalogs and tables for AWS Glue Data Catalogs.

Column	Description
DCAT_CON_ID	Unique identifier of the data catalog. The connection id. **OCI Data Catalog example**: `CON1` **AWS Glue Data Catalog example**: `CON1`

Column	Description
CATALOG_ID	Data catalog unique identifier. **OCI Data Catalog example**: ocid1.datacatalog.oc1.ap-mumbai-1.….y35a **AWS Glue Data Catalog example**: NULL 579294766787
DATABASE_NAME	Name of the data asset (OCI)/ database (AWS Glue). **OCI Data Catalog example**: OBJECT_STORE_AT_ASHBURN **AWS Glue Data Catalog example**: OBJECT_STORE_AT_N_CALIFORNIA
NAME	Name of the data entity (OCI) / table (AWS Glue). **OCI Data Catalog example**: BIKES_TRIPS **AWS Glue Data Catalog example**: BIKES_TRIPS
DESCRIPTION	Description of the data entity (OCI) / table (AWS Glue). **OCI Data Catalog example**: Table storing bike trips **AWS Glue Data Catalog example**: Table storing bike trips
TIME_CREATED	The date and time the data entity (OCI) / table (AWS Glue) was created in the data catalog. **OCI Data Catalog example**: 26-SEP-22 10.56.01.395000 PM +00:00 **AWS Glue Data Catalog example**: 2022-06-15T09:45:35+01:00
TIME_UPDATED	Last time a change was made to the data entity (OCI) / table (AWS Glue). **OCI Data Catalog example**: 26-SEP-22 10.56.01.395000 PM +00:00 **AWS Glue Data Catalog example**: 2022-06-15T09:45:35+01:00

Column	Description
DETAILS	JSON document with metadata about each each data entity (OCI) / table (AWS Glue) **OCI Data Catalog example**: `{` `  "business-name": null,` `  "data-asset-key": "bc95181c-3ac3-4959-9e5f-...",` `  "description": null,` `  "display-name": "bikes_trips",` `  "external-key": "LE: https://swiftobjectstorage.us-ashburn-1.oraclecloud.com/v1/..._trips",` `  "folder-key": "9c4b542d-d6eb-4b83-bf59-...",` `  "folder-name": "hive",` `  "is-logical": true,` `  "is-partition": false,` `  "key": "fde30a69-a07c-478a-ab62-...",` `  "lifecycle-state": "ACTIVE",` `  "object-storage-url": "https://objectstorage.us-ashburn-1.oraclecloud.com/n/...",` `  "path": "OBJECT_STORE_AT_ASHBURN/hive/hive",` `  "pattern-key": "db21b3f1-1508-4045-aa80-...",` `  "properties": {` `  "default": {` `  "CONTENT-LENGTH": "4310321",` `  "LAST-MODIFIED": "Fri, 9 Oct 2020 20:16:52 UTC",` `  "archivedPECount": "0",` `  "dataEntityExpression": "{logicalEntity:[^/]+}.db/{logicalEntity:[^/]+}/.*",` `  "harvestedFile": "bikes.db/trips/p_start_month=2019-09/000000_0",` `  "patternName": "bikes_trips"` `  },` `  "harvestProps": {` `  "characterset": "UTF8",` `  "compression": "none",` `  "type": "PARQUET"` `  }` `  },` `  "realized-expression": "bikes.db/trips/.*",` `  "time-created": "2022-09-26T22:56:35.063000+00:00",` `  "time-updated": "2022-09-26T22:56:35.063000+00:00",` `  "type-key": "6753c3af-7f88-44b9-be52-1d57bef462fb",` `  "updated-by-id": "ocid1.user.oc1..r5l3tov7a",` `  "uri": "/dcat/20190325/dataAssets/bc95181c-3ac3-4959-9e5f-..."` `}` **AWS Glue Data Catalog example**: `{` `  "Name": "bikes_trips",`

Column	Description


```
    "DatabaseName": "dbmsdcatpoc",
    "Owner": "owner",
    "CreateTime": "2022-06-23T13:24:20+01:00",
    "UpdateTime": "2022-06-23T13:24:20+01:00",
    "LastAccessTime": "2022-06-23T13:24:20+01:00",
    "Retention": 0,
    "StorageDescriptor": {
        "Columns": [
            {
                "Name": "trip_duration",
                "Type": "int"
            },
            {
                "Name": "start_month",
                "Type": "string"
            }, ...
        ],
        "Location": "s3://dbmsdcatpoc/hive/bikes.db/trips/",
        "InputFormat":
"org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputForma
t",
        "OutputFormat":
"org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputForm
at",
        "Compressed": false,
        "NumberOfBuckets": -1,
        "SerdeInfo":
 {            "SerializationLibrary":
"org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe"
,
            "Parameters": {
                "serialization.format": "1"
            }
        },
        "BucketColumns": [],
        "SortColumns": [],
        "Parameters": {
            "CrawlerSchemaDeserializerVersion": "1.0",
            "CrawlerSchemaSerializerVersion": "1.0",
            "UPDATED_BY_CRAWLER": "crawler-bikes",
            "averageRecordSize": "86",
            "classification": "parquet",
            "compressionType": "none",
            "objectCount": "12",
            "recordCount": "404947",
            "sizeKey": "35312159",
            "typeOfData": "file"
        },
        "StoredAsSubDirectories": false
    },
    "PartitionKeys": [
```

Column	Description
	{ "Name": "p_start_month", "Type": "string" }], "TableType": "EXTERNAL_TABLE", "Parameters": { "CrawlerSchemaDeserializerVersion": "1.0", "CrawlerSchemaSerializerVersion": "1.0", "UPDATED_BY_CRAWLER": "crawler-bikes", "averageRecordSize": "86", "classification": "parquet", "compressionType": "none", "objectCount": "12", "recordCount": "404947", "sizeKey": "35312159", "typeOfData": "file" }, "CreatedBy": "arn:aws:sts::579294766787:assumed-role/ AWSGlueServiceRole-dbmsdcat/AWS-Crawler", "IsRegisteredWithLakeFormation": false, "CatalogId": "579294766787", "VersionId": "0" }

Example

ALL_DCAT_ASSETS View

The Data Catalog assets that this database is authorized to access.

Column	Datatype	Description
DCAT_CON_ID	VARCHAR2 (4000)	Connection identifier that is unique within the instance
KEY	VARCHAR2(4000)	Asset key
DISPLAY_NAME	VARCHAR2(4000)	Asset display name
DESCRIPTION	VARCHAR2(4000)	Asset description
CATALOG_ID	VARCHAR2(4000)	OCID for the Data Catalog containing the asset
EXTERNAL_KEY	VARCHAR2(4000)	Base Object Storage URI for the asset
URI	VARCHAR2(4000)	Asset URI for the Data Catalog API
TIME_CREATED	TIMESTAMP(6) WITH TIMEZONE	The date and time the data asset was created
TYPE_KEY	VARCHAR2(4000)	The key of the data asset type (currently, only Object Storage data assets are supported). Type keys can be found via the '/types' Data Catalog endpoint.

Column	Datatype	Description
LIFECYCLE_STATE	VARCHAR2(4000)	The current state of the data asset. For more information on possible life cycle states, see the Data Catalog DataAsset Reference for a list of possible states for `lifecycleState`.

ALL_DCAT_ATTRIBUTES View

The Data Catalog attributes this database is authorized to access.

Column	Datatype	Description
DCAT_CON_ID	VARCHAR2 (4000)	Connection identifier that is unique within the instance
KEY	NUMBER	Attribute key
DISPLAY_NAME	VARCHAR2(4000)	Attribute display name
BUSINESS_NAME	VARCHAR2(4000)	Attribute business name
DESCRIPTION	VARCHAR2(4000)	Attribute description
DATA_ASSET_KEY	VARCHAR2(4000)	Data asset key
FOLDER_KEY	VARCHAR2(4000)	Folder key
ENTITY_KEY	VARCHAR2(4000)	Entity key
EXTERNAL_KEY	VARCHAR2(4000)	Unique external key for the attribute
LENGTH	NUMBER	Maximum allowed length of the attribute value
PRECISION	NUMBER	Precision of the attribute value (usually applies to float data type)
SCALE	NUMBER	Scale of the attribute value (usually applies to float data type)
IS_NULLABLE	NUMBER	Identifies if this attribute can be assigned null values
URI	VARCHAR2(4000)	URI to the attribute instance in the Data Catalog API
LIFECYCLE_STATE	VARCHAR2(4000)	The current state of the attribute. For more information on possible life cycle states, see the Data Catalog Attribute Reference for a list of possible states for `lifecycleState`.
TIME_CREATED	TIMESTAMP(6) WITH TIME ZONE	The date and time the attribute was created
EXTERNAL_DATA_TYPE	VARCHAR2(4000)	Data type of the attribute as defined in the external system
MIN_COLLECTION_COUNT	NUMBER	Minimum number of elements, if the type of the attribute is a collection type
MAX_COLLECTION_COUNT	NUMBER	Maximum number of elements, if the type of the attribute is a collection type
DATATYPE_ENTITY_KEY	VARCHAR2(4000)	Entity key that represents the datatype of this attribute, applicable if this attribute is a complex type
EXTERNAL_DATATYPE_ENTITY_KEY	VARCHAR2(4000)	External entity key that represents the datatype of this attribute, applicable if this attribute is a complex type

Column	Datatype	Description
PARENT_ATTRIBUTE_KEY	VARCHAR2(4000)	Attribute key that represents the parent attribute of this attribute, applicable if the parent attribute is of complex datatype
EXTERNAL_PARENT_ATTRIBUTE_KEY	VARCHAR2(4000)	External attribute key that represents the parent attribute of this attribute, applicable if the parent attribute is of complex type
PATH	VARCHAR2(4000)	Full path of the attribute

ALL_DCAT_CONNECTIONS View

A view that contains information about the data catalog(s) connected to this instance.

Column	Datatype	Description
DCAT_CON_ID	VARCHAR2(4000)	Connection identifier that is unique within the instance
COMPARTMENT_ID	VARCHAR2(4000)	OCID for the compartment where the Data Catalog instance resides
INSTANCE_ID	VARCHAR2(4000)	OCID for the Data Catalog instance
REGION	VARCHAR2(4000)	Region for the Data Catalog instance
ENDPOINT	VARCHAR2(4000)	Endpoint for the Data Catalog instance
CREATED	TIMESTAMP	When the Data Catalog instance was created
NAME	VARCHAR2(4000)	Name of the Data Catalog instance
LAST_UPDATED	TIMESTAMP	Timestamp of the last update of the connection to the Data Catalog instance
LATEST_OPERATION_ID	NUMBER	The id of the last synchronization operation
DATA_CATALOG_CREDENTIAL	VARCHAR2(128)	Credential used for accessing the Data Catalog
OBJECT_STORE_CREDENTIAL	VARCHAR2(128)	Credential used by the external table driver for accessing the Object Store

ALL_DCAT_ENTITIES View

The Data Catalog logical entities this database is authorized to access.

Column	Datatype	Description
DCAT_CON_ID	VARCHAR2(4000)	Connection identifier that is unique within the instance
CATALOG_ID	VARCHAR2(4000)	OCID for the Data Catalog containing the asset
KEY	VARCHAR2(4000)	Entity key
DISPLAY_NAME	VARCHAR2(4000)	Entity display name
BUSINESS_NAME	VARCHAR2(4000)	Entity business name
DESCRIPTION	VARCHAR2(4000)	Logical entity description
DATA_ASSET_KEY	VARCHAR2(4000)	Asset key
FOLDER_KEY	VARCHAR2(4000)	Folder unique key
FOLDER_NAME	VARCHAR2(4000)	Folder name (bucket)

Column	Datatype	Description
EXTERNAL_KEY	VARCHAR2(4000)	External key for the logical entity
PATTERN_KEY	VARCHAR2(4000)	Key of the associated pattern for the logical entity
REALIZED_EXPRESSION	VARCHAR2(4000)	The regular expression used to obtain the files for this logical entity
PATH	VARCHAR2(4000)	Full path for the logical entity
TIME_CREATED	TIMESTAMP(6) WITH TIME ZONE	Date and time the entity was created
TIME_UPDATED	TIMESTAMP(6) WITH TIME ZONE	Last time a change was made to the data entity
UPDATED_BY_ID	VARCHAR2(4000)	OCID of the user who updated this object in the Data Catalog
URI	VARCHAR2(4000)	URI of the entity instance in the API
LIFECYCLE_STATE	VARCHAR2(4000)	The current state of the entity. For more information on possible life cycle states, see the Data Catalog Entity Reference for a list of possible states for `lifecycleState`.

ALL_DCAT_FOLDERS View

Metadata for the Object Storage buckets containing the data files for the Logical Entities.

Column	Datatype	Description
DCAT_CON_ID	VARCHAR2(4000)	Connection identifier that is unique within the instance
CATALOG_ID	VARCHAR2(4000)	OCID for the Data Catalog containing the asset
KEY	VARCHAR2(4000)	Folder key
DISPLAY_NAME	VARCHAR2(4000)	Folder display name
BUSINESS_NAME	VARCHAR2(4000)	Folder business name
DESCRIPTION	VARCHAR2(4000)	Folder description
DATA_ASSET_KEY	VARCHAR2(4000)	Key for the data asset containing the folder
PARENT_FOLDER_KEY	VARCHAR2(4000)	Key for the parent folder (currently, this is the data asset key)
PATH	VARCHAR2(4000)	Full path for the folder
EXTERNAL_KEY	VARCHAR2(4000)	Object Storage URI for the bucket
TIME_EXTERNAL	TIMESTAMP(6) WITH TIMEZONE	The last modified timestamp of this folder
TIME_CREATED	TIMESTAMP(6) WITH TIMEZONE	The date/time the folder was created
URI	VARCHAR2(4000)	URI to the folder instance in the Data Catalog API.
LIFECYCLE_STATE	VARCHAR2(4000)	The current state of the folder. For more information on possible life cycle states, see the Data Catalog Folder Reference for a list of possible states for `lifecycleState`.

ALL_DCAT_GLOBAL_ACCESSIBLE_CATALOGS View

This view lists all accessible catalogs across all regions, along with the level of access privileges for each catalog.

Column	Datatype	Description
CATALOG_ID	VARCHAR2(4000)	Catalog OCID
CATALOG_NAME	VARCHAR2(4000)	Name of the catalog
CATALOG_REGION	VARCHAR2(4000)	Name of the catalog region
CATALOG_SCORE	NUMBER	The catalog score is a numeric value calculated from the privileges configured for the Data Catalog access credential. A higher catalog score means greater privileges, which may equate to a higher likelihood that this catalog is intended for use with this Autonomous Database instance.

ALL_DCAT_LOCAL_ACCESSIBLE_CATALOGS View

This view lists all accessible catalogs in the current region, along with the level of access privileges for each catalog.

Column	Datatype	Description
CATALOG_ID	VARCHAR2(4000)	Catalog OCID
CATALOG_NAME	VARCHAR2(4000)	Name of the catalog
CATALOG_SCORE	NUMBER	The catalog score is a numeric value calculated from the privileges configured for the Data Catalog access credential. A higher catalog score means greater privileges, which may equate to a higher likelihood that this catalog is intended for use with this Autonomous Database instance.

ALL_GLUE_DATABASES View

The AWS Glue Data Catalog databases that the data catalog credential is authorized to access.

Column	Data Type	Description
DCAT_CON_ID	VARCHAR2(4000)	Unique identifier of data catalog connection id.
CATALOG_ID	VARCHAR2(255)	Data Catalog unique identifier.
NAME	VARCHAR2(255)	Name of the database.
DESCRIPTION	VARCHAR2(2048)	Description of the database.
LOCATION_URI	VARCHAR2(1024)	The location of the database.
CREATE_TIME	TIMESTAMP	The time that the database was created in the data catalog.

Column	Data Type	Description
PARAMETERS	CLOB	JSON document with key-value pairs that define parameters and properties of the database.
TARGET_DATABASE	VARCHAR2(4000)	JSON document that describes a target database for resource linking in AWS.

ALL_GLUE_TABLES View

This view shows all AWS Glue Data Catalog tables that the data catalog credential is authorized to access.

Column	Data Type	Description
DCAT_CON_ID	VARCHAR2(4000)	Unique identifier of data catalog connection id.
CATALOG_ID	VARCHAR2(255)	Catalog identifier
DATABASE_NAME	VARCHAR2(255)	Database name
NAME	VARCHAR2(255)	Table name
TABLE_TYPE	VARCHAR2(255)	Table type
CLASSIFICATION	VARCHAR2(255)	
DESCRIPTION	VARCHAR2(2048)	Table description
OWNER	VARCHAR2(255)	Table owner
CREATED_BY	VARCHAR2(255)	Table creator
CREATE_TIME	TIMESTAMP	The time the table was created in the data catalog.
LAST_ANALYZED_TIME	TIMESTAMP	The last time column statistics were computed for this table.
LAST_ACCESS_TIME	TIMESTAMP	The last time the table was accessed.
UPDATE_TIME	TIMESTAMP	The last time the table was updated.
IS_REGISTERED_WITH_LAKE_FORMATION	NUMBER	Indicates whether the table is registered with AWS lake formation.
PARAMETERS	CLOB	JSON document with key-value pairs that define properties of the table.
PARTITION_KEYS	CLOB	JSON document with a list of columns by which the table is partitioned.
RETENTION	NUMBER	The retention time for this table.
STORAGE_DESCRIPTION	CLOB	JSON document with information about the physical storage of a table.
TARGET_TABLE	VARCHAR2(4000)	JSON document describing a target table used for resource linking in AWS.
VERSION_ID	VARCHAR2(255)	The version identifier for the table.
VIEW_EXPANDED_TEXT	CLOB	Introduced by AWS Glue for compatibility with Hive. Not used by AWS Glue.
VIEW_ORIGINAL_TEXT	CLOB	Introduced by AWS Glue for compatibility with Hive. Not used by AWS Glue.

DCAT_ATTRIBUTES View

Lists the mapping of logical entity attributes to external table columns.

Column	Datatype	Description
DCAT_CON_ID	VARCHAR2 (4000)	Connection identifier that is unique within the instance
ASSET_KEY	VARCHAR2(4000)	Data Catalog asset key
ENTITY_KEY	VARCHAR2(4000)	Data Catalog entity key
ATTRIBUTE_KEY	VARCHAR2(4000)	Data Catalog attribute key
ORACLE_COLUMN_NAME	VARCHAR2(128)	Mapped column name

DCAT_ENTITIES View

Describes the mapping of logical entities to external tables.

Column	Datatype	Description
DCAT_CON_ID	VARCHAR2(4000)	Connection identifier that is unique within the instance
ASSET_KEY	VARCHAR2(4000)	Data Catalog asset key
ENTITY_KEY	VARCHAR2(4000)	Data Catalog entity key
FOLDER_KEY	VARCHAR2(4000)	Data Catalog folder key
ORACLE_TABLE_NAME	VARCHAR2(128)	Mapped table name
ORACLE_SCHEMA_NAME	VARCHAR2(128)	Mapped schema name
ENTITY_ORACLE_DB_SCHEMA	VARCHAR2(4000)	The entity's oracle-db-schema custom property used to derive the schema
ASSET_ORACLE_DB_SCHEMA	VARCHAR2(4000)	The data asset's oracle-db-schema custom property used to derive the schema
FOLDER_ORACLE_DB_SCHEMA	VARCHAR2(4000)	The folder's oracle-db-schema custom property used to derive the schema

DBMS_DCAT$SYNC_LOG View

The DBMS_DCAT$SYNC_LOG view provides easy access to the log table for the last sync operation executed by the current user.

Every call to the RUN_SYNC procedure is logged to a new log table, pointed to by the LOGFILE_TABLE field of USER_LOAD_OPERATIONS. The log tables are automatically dropped after 2 days, and users can clear all sync logs using the DBMS_CLOUD.DELETE_ALL_OPERATIONS procedure where type is DCAT_SYNC.

The DBMS_DCAT$SYNC_LOG view automatically identifies the latest log table. The schema for the DBMS_DCAT$SYNC_LOG view is described below and the access permissions are identical to those of the individual log tables. By default READ is granted to the dbms_dcat role and to the ADMIN user.

The log tables have the following format:

Column	Datatype	Description
LOG_TIMESTAMP	TIMESTAMP	Timestamp for the log entry.
LOG_LEVEL	VARCHAR2(32)	The entry log level can have one of the following values: OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE, ALL.
LOG_DETAILS	VARCHAR2(32767)	The log message.

DBMS_MAX_STRING_SIZE Package

The DBMS_MAX_STRING_SIZE package provides an interface for checking and changing the value of the DBMS_MAX_STRING_SIZE initialization parameter.

- CHECK_MAX_STRING_SIZE Function
  This function checks if the MAX_STRING_SIZE parameter can be updated to a given value and returns a list of violations that would prevent the parameter from being updated.
- MODIFY_MAX_STRING_SIZE Procedure
  This procedure updates the value of the MAX_STRING_SIZE parameter to a given value.

CHECK_MAX_STRING_SIZE Function

This function checks if the MAX_STRING_SIZE parameter can be updated to a given value and returns a list of violations that would prevent the parameter from being updated.

Syntax

```
DBMS_MAX_STRING_SIZE.CHECK_MAX_STRING_SIZE(
       new_value    IN VARCHAR2)
RETURN DBMS_MAX_STRING_SIZE_TBL;
```

Parameters

Parameter	Description
new_value	Specifies the new MAX_STRING_SIZE parameter value to be set. The only valid value is: 'STANDARD'.

Usage Notes

If the return list is empty, then there are no violations and the MAX_STRING_SIZE update can be performed.

Example

```
SELECT * FROM TABLE(DBMS_MAX_STRING_SIZE.CHECK_MAX_STRING_SIZE('STANDARD'));

TYPE   OBJECT_OWNER OBJECT_NAME COLUMN_NAME
REASON
------ ------------ ----------- --------------
-------------------------------------------
COLUMN ADMIN        SALES       CUST_NOTES     Physical column exceeds
```

```
STANDARD length limit
1 rows selected.
```

MODIFY_MAX_STRING_SIZE Procedure

This procedure updates the value of the `MAX_STRING_SIZE` parameter to a given value.

Syntax

```
DBMS_MAX_STRING_SIZE.MODIFY_MAX_STRING_SIZE(
          new_value    IN VARCHAR2);
```

Where: *user_account* is the user account name (schema name).

Parameters

Parameter	Description
`new_value`	Specifies the new `MAX_STRING_SIZE` parameter value to be set. The only valid value is: `'STANDARD'`.

Usage Notes

- Using `DBMS_MAX_STRING_SIZE.MODIFY_MAX_STRING_SIZE` is a one-way change that cannot be reverted. After a database is switched back to the `STANDARD` style of supporting a maximum length of 4000 bytes for the `VARCHAR2`, `NVARCHAR2`, and `RAW` data types, you cannot re-enable `EXTENDED` data types.
- The ADMIN user is granted `EXECUTE` privilege `WITH GRANT OPTION` clause on `DBMS_MAX_STRING_SIZE`. Oracle recommends that you do not `GRANT EXECUTE` on this package to other users.
- The error `ORA-20000` is raised if any object exists that would prevent `MAX_STRING_SIZE` from being updated.
- The ADMIN user is granted `EXECUTE` privilege `WITH GRANT OPTION` clause on `DBMS_MAX_STRING_SIZE`. Oracle recommends that you do not `GRANT EXECUTE` on this package to other users.

Example

```
SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME = 'max_string_size';

NAME                 VALUE
max_string_size      EXTENDED


   BEGIN
     DBMS_MAX_STRING_SIZE.MODIFY_MAX_STRING_SIZE('STANDARD');
   END;
/


PL/SQL procedure successfully completed.
```

```
SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME = 'max_string_size';

NAME                 VALUE
max_string_size      STANDARD
```

DBMS_AUTO_PARTITION Package

The `DBMS_AUTO_PARTITION` package provides administrative routines for managing automatic partitioning of schemas and tables.

- CONFIGURE Procedure
  This procedure configures settings for automatic partitioning in Autonomous Database.
- VALIDATE_CANDIDATE_TABLE Function
  This function checks if the given table is a valid candidate for automatic partitioning in Autonomous Database.
- RECOMMEND_PARTITION_METHOD Function
  This function returns a recommendation ID that can be used with `APPLY_RECOMMENDATION` procedure to apply the recommendation, or can be used with `DBA_AUTO_PARTITION_RECOMMENDATIONS` view to retrieve details of the recommendations for automatic partitioning in Autonomous Database.
- APPLY_RECOMMENDATION Procedure
  This procedure applies the given recommendation in an Autonomous Database.
- REPORT_ACTIVITY Function
  This function returns a report of the automatic partitioning operations executed during a specific period in an Autonomous Database.
- REPORT_LAST_ACTIVITY Function
  This function returns a report of the most recent automatic partitioning operation executed in an Autonomous Database.

CONFIGURE Procedure

This procedure configures settings for automatic partitioning in Autonomous Database.

Syntax

```
DBMS_AUTO_PARTITION.CONFIGURE (
     PARAMETER_NAME     IN VARCHAR2,
     PARAMETER_VALUE    IN VARCHAR2,
     ALLOW              IN BOOLEAN    DEFAULT TRUE);
```

Parameters

Parameter	Description
PARAMETER_NAME	Name of the automatic partitioning configuration parameter to update. It can have one of the following values: • AUTO_PARTITION_MODE • AUTO_PARTITION_SCHEMA • AUTO_PARTITION_TABLE • AUTO_PARTITION_REPORT_RETENTION AUTO_PARTITION_MODE sets the mode of automatic partitioning operation, and has one of the following values: • IMPLEMENT: In this mode, automatic partitioning generates a report and modifies the existing table using the recommended partition method. • REPORT ONLY: In this mode, automatic partitioning generates a report but existing tables are not modified. This is the default value. • OFF: In this mode, automatic partitioning is prevented from generating, considering, or applying recommendations. It does not disable existing automatic partitioned tables. AUTO_PARTITION_SCHEMA sets schemas to include or exclude from using automatic partitioning. Its behavior is controlled by the allow parameter. The automatic partitioning process manages two schema lists. 1. Inclusion list is the list of schemas, case-sensitive, that can use automatic partitioning. 2. Exclusion list is the list of schemas, case-sensitive, that cannot use automatic partitioning. Initially, both lists are empty, and all schemas in the database can use automatic partitioning. If the inclusion list contains one or more schemas, then only the schemas listed in the inclusion list can use automatic partitioning. If the inclusion list is empty and the exclusion list contains one or more schemas, then all schemas use automatic partitioning except the schemas listed in the exclusion list. If both lists contain one or more schemas, then all schemas use automatic partitioning except the schemas listed in the exclusion list. AUTO_PARTITION_TABLE sets tables to include or exclude from using auto partitioning. The parameter value is <schema_name>.<table_name>. The automatic partitioning process manages two table lists. 1. Inclusion list is the list of tables, case-sensitive, that can use automatic partitioning. 2. Exclusion list is the list of tables, case-sensitive, that cannot use automatic partitioning. Initially, both lists are empty, and all tables in the database can use automatic partitioning. If the inclusion list contains one or more tables, then only the tables listed in the inclusion list can use automatic partitioning. If the inclusion list is empty and the exclusion list contains one or more tables, then all tables use automatic partitioning except the tables listed in the exclusion list. If both lists contain one or more tables, then all tables use automatic partitioning except the tables listed in the exclusion list. If a table is not on either list, the schema inclusion and exclusion lists decide if a table is a candidate table for automatic partitioning. If there is a conflict between the schema level lists and the table level lists, the table level lists take precedence.

Parameter	Description
	To remove all tables from inclusion and exclusion lists run: `DBMS_AUTO_PARTITION.CONFIGURE('AUTO_PARTITION_TABLE', NULL);` `AUTO_PARTITION_REPORT_RETENTION` sets the number of days for which automatic partitioning logs are retained in the database before they are deleted. An automatic partitioning report cannot be generated for a period beyond the value specified for this value. Default value is 90 days.
`PARAMETER_VALUE`	Value for the configuration setting specified in `parameter_name`. When set to `NULL`, the configuration setting is assigned its default value.
`ALLOW`	Applicable only for the `AUTO_PARTITION_SCHEMA` or `AUTO_PARTITION_TABLE` configuration settings with one of the following values: • `TRUE` adds specified schema or table to the inclusion list. • `FALSE` removes specified schema or table from the exclusion list. • `NULL` removes specified schema or table from the list to which currently assigned. Refer to the description of the `AUTO_PARTITION_SCHEMA` and `AUTO_PARTITION_TABLE` configuration settings for more information about inclusion lists and exclusion lists.

Usage Notes

- You can check the current setting for automatic partitioning configuration using the following SQL:

  ```
  SELECT * FROM DBA_AUTO_PARTITION_CONFIG;
  ```

- Unlike automatic indexing, automatic partitioning does not run periodically as a background task. Automatic partitioning only runs when you invoke it using the `DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD` function.

VALIDATE_CANDIDATE_TABLE Function

This function checks if the given table is a valid candidate for automatic partitioning in Autonomous Database.

Valid Candidate

To be a valid candidate, the following tests must pass:

- Table passes inclusion and exclusion tests specified by `AUTO_PARTITION_SCHEMA` and `AUTO_PARTITION_TABLE` configuration parameters.
- Table exists and has up-to-date statistics.
- Table is at least 64 GB.
- Table has 5 or more queries in the SQL tuning set that scanned the table.
- Table does not contain a `LONG` data type column.
- Table is not manually partitioned.

- Table is not an external table, an internal/external hybrid table, a temporary table, an index-organized table, or a clustered table.
- Table does not have a domain index or bitmap join index.
- Table is not an advance queuing, materialized view, or flashback archive storage table.
- Table does not have nested tables, or certain other object features.

Returns:

- `VALID` if the table is a valid candidate for autonomous partitioning
- `INVALID: <reason>` if the table is not a valid candidate for autonomous partitioning, and <reason> is a string describing why the table is not a valid candidate.

Syntax

```
DBMS_AUTO_PARTITION.VALIDATE_CANDIDATE_TABLE
    ( SQLSET_OWNER  IN VARCHAR2   DEFAULT 'SYS',
      SQLSET_NAME   IN VARCHAR2   DEFAULT 'SYS_AUTO_STS',
      TABLE_OWNER   IN VARCHAR2,
      TABLE_NAME    IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Parameter	Description
`SQLSET_OWNER`, `SQLSET_NAME`	Name of SQL tuning set representing the workload to be evaluated.
`TABLE_OWNER`, `TABLE_NAME`	Name of a table to validate as a candidate for automatic partitioning.

Usage Notes

- As an example, you can check the validity of a sample table, `LINEORDER` in schema `TEST`, with the following SQL:

```
SELECT DBMS_AUTO_PARTITION.VALIDATE_CANDIDATE_TABLE
    ( TABLE_OWNER => 'TEST',
      TABLE_NAME  => 'LINEORDER')
FROM DUAL;
```

RECOMMEND_PARTITION_METHOD Function

This function returns a recommendation ID that can be used with `APPLY_RECOMMENDATION` procedure to apply the recommendation, or can be used with `DBA_AUTO_PARTITION_RECOMMENDATIONS` view to retrieve details of the recommendations for automatic partitioning in Autonomous Database.

Syntax

```
DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD
    ( SQLSET_OWNER     IN VARCHAR2   DEFAULT 'SYS',
      SQLSET_NAME      IN VARCHAR2   DEFAULT 'SYS_AUTO_STS',
      TABLE_OWNER      IN VARCHAR2   DEFAULT NULL,
```

```
    TABLE_NAME       IN VARCHAR2   DEFAULT NULL,
    TIME_LIMIT       IN INTERVAL DAY TO SECOND DEFAULT INTERVAL '1' DAY,
    REPORT_TYPE      IN VARCHAR2   DEFAULT 'TEXT',
    REPORT_SECTION   IN VARCHAR2   DEFAULT 'SUMMARY',
    REPORT_LEVEL     IN VARCHAR2   DEFAULT 'TYPICAL')
 RETURN RAW;
```

Parameters

Parameter	Description
SQLSET_OWNER, SQLSET_NAME	Name of SQL tuning set representing the workload to be evaluated.
TABLE_OWNER, TABLE_NAME	Name of a table to validate as a candidate for automatic partitioning.
TIME_LIMIT	When the function chooses the tables for which to generate recommendations, TABLE_OWNER and TABLE_NAME are NULL), parameter limits how long the function runs before it stops looking for new candidate tables to partition. Once started processing a table, process will not terminate. It is expected that the function may run longer than this parameter. If this parameter is NULL there is no time limit. The default is 1 day.
REPORT_TYPE	Used to generate report for recommended partition method. See REPORT_ACTIVITY Function for details.
REPORT_SECTION	Used to generate persistent report for recommended partition method. See REPORT_ACTIVITY Function for details.
REPORT_LEVEL	Used to generate report for recommended partition method. See REPORT_ACTIVITY Function for details.

Usage Notes

- The AUTO_PARTITION_MODE controls the actions taken by this function:
  - IMPLEMENT: In this mode, automatic partitioning generates a report and modifies the existing table using the recommended partition method.
  - REPORT ONLY: In this mode, automatic partitioning generates a report generated but existing tables are not modified. This is the default value.
  - OFF: In this mode, automatic partitioning prevented from producing, considering, or applying new recommendations. It does not disable existing automatic partitioned tables.
- Unlike automatic indexing, automatic partitioning does not run periodically as a background task. Automatic partitioning only runs when you invoke it using the DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD function.

Return Values

This function returns a recommendation ID that can be used as follows: DBMS_AUTO_PARTITION.APPLY_RECOMMENDATION to apply the recommendation,

- Use with DBMS_AUTO_PARTITION.APPLY_RECOMMENDATION to apply the recommendation.

- Use with `DBA_AUTO_PARTITION_RECOMMENDATIONS` view to retrieve details of the recommendations. For example:

```
SELECT PARTITION_METHOD, PARTITION_KEY
     FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
     WHERE RECOMMENDATION_ID = :RECOMMENDATION_ID;
```

APPLY_RECOMMENDATION Procedure

This procedure applies the given recommendation in an Autonomous Database.

Syntax

```
DBMS_AUTO_PARTITION.APPLY_RECOMMENDATION
    ( RECOMMENDATION_ID  IN RAW,
      TABLE_OWNER        IN VARCHAR2   DEFAULT NULL,
      TABLE_NAME         IN VARCHAR2   DEFAULT NULL);
```

Parameters

Parameter	Description
`RECOMMENDATION_ID`	Recommendation ID returned from `RECOMMEND_PARTITION_METHOD` function or queried from `DBA_AUTO_PARTITION_RECOMMENDATIONS` view.
`TABLE_OWNER`, `TABLE_NAME`	When a single recommendation ID has recommendations for multiple tables, this optional parameter allows you to control which tables are partitioned. • If parameters are NULL, partition all tables recommended in the given recommendation ID. • If a table name is given, partition only the named table. • If either `TABLE_OWNER` or `TABLE_NAME` is `NOT NULL`, they must both be `NOT NULL`.

Usage Note:

Regardless of `AUTO_PARTITION_MODE`, this procedure raises an `ORA-20000: recommendation_id was not found` if either there are no accepted recommendations associated with the `RECOMMENDATION_ID`, or all accepted recommendations associated with the `RECOMMENDATION_ID` have already been applied. The first case applies if `RECOMMENDATION_ID` was generated with `AUTO_PARTITION_MODE = OFF`. The second case applies if `RECOMMENDATION_ID` was generated with `AUTO_PARTITION_MODE = IMPLEMENT`.

REPORT_ACTIVITY Function

This function returns a report of the automatic partitioning operations executed during a specific period in an Autonomous Database.

Syntax

```
DBMS_AUTO_PARTITION.REPORT_ACTIVITY
    ( ACTIVITY_START     IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
      ACTIVITY_END       IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
      TYPE               IN VARCHAR2                 DEFAULT 'TEXT',
```

```
      SECTION           IN VARCHAR2               DEFAULT 'ALL',
      LEVEL             IN VARCHAR2               DEFAULT 'TYPICAL')
RETURN CLOB;
```

Parameters

Parameter	Description
ACTIVITY_START	Starting time automatic partitioning operations use for the report. If no value is specified, or NULL is specified, the report is generated for the last automatic partitioning operation that was executed.
ACTIVITY_END	Ending time automatic partitioning operations use for the report. If no value is specified, or NULL is specified, then the report is generated for the last automatic partitioning operation that was executed.
TYPE	Format of the report that has one of the following values: • TEXT (default) • HTML • XML
SECTION	Sections to include in the report that has one of the following values: • SUMMARY - Include only the workload summary in the report • ALL = Include all the sections in the report. (default)
level	Level of information to include in the report that has one of the following values: • TYPICAL - Include typical automatic partitioning information in the report (default). • CHANGED - Include only SQL with changed performance in the report. • IMPROVED - Include only SQL with improved performance in the report. • REGRESSED - Include only SQL with regressed performance in the report. • UNCHANGED - Include only SQL with unchanged performance in the report. • ALL - Include all automatic partitioning information in the report.

Usage Notes

Returns: A performance analysis report for workload executed on database after recommendation is applied. This report is not stored persistently with the recommendation.

REPORT_LAST_ACTIVITY Function

This function returns a report of the most recent automatic partitioning operation executed in an Autonomous Database.

Syntax

```
DBMS_AUTO_PARTITION.REPORT_LAST_ACTIVITY
    ( TYPE    IN VARCHAR2 DEFAULT 'TEXT',
      SECTION IN VARCHAR2 DEFAULT 'ALL',
      LEVEL   IN VARCHAR2 DEFAULT 'TYPICAL')
RETURN CLOB;
```

Parameters

Parameter	Description
TYPE	The output format of the report, see REPORT_ACTIVITY Function for information.
SECTION	The sections included in the report, see REPORT_ACTIVITY Function for information.
LEVEL	The level of information included in the report, see REPORT_ACTIVITY Function for information.

Usage Notes

Returns: A performance analysis report for workload executed on database after latest recommendation is applied. This report is not stored persistently with the recommendation.

CS_RESOURCE_MANAGER Package

The CS_RESOURCE_MANAGER package provides an interface to list and update consumer group parameters, and to revert parameters to default values.

- LIST_CURRENT_RULES Function
  This function lists the parameter values for each consumer group.
- LIST_DEFAULT_RULES Function
  This function returns the default values for all consumer groups.
- REVERT_TO_DEFAULT_VALUES Procedure
  This procedure reverts the specified resource manager's plan properties to default values.
- UPDATE_PLAN_DIRECTIVE Procedure
  Use this procedure to update the resource plan for a specified consumer group.

LIST_CURRENT_RULES Function

This function lists the parameter values for each consumer group.

Syntax

```
CS_RESOURCE_MANAGER.LIST_CURRENT_RULES
   RETURN TABLE;
```

Example

```
SELECT * FROM CS_RESOURCE_MANAGER.LIST_CURRENT_RULES();

CONSUMER_GROUP ELAPSED_TIME_LIMIT IO_MEGABYTES_LIMIT SHARES CONCURRENCY_LIMIT
DEGREE_OF_PARALLELISM
-------------- ------------------ ------------------ ------ -----------------
--------------------
HIGH                                                     4
3                    3
MEDIUM                                                   2
2                    9
```

```
LOW                                                          1
900                     1
```

LIST_DEFAULT_RULES Function

This function returns the default values for all consumer groups.

Syntax

```
CS_RESOURCE_MANAGER.LIST_DEFAULT_RULES
  RETURN TABLE;
```

Usage Note

- By default the parallel degree policy value is `MANUAL` for the `TPURGENT` consumer group. The `CS_RESOURCE_MANAGER.LIST_DEFAULT_RULES` function shows no value for the default value for the `DEGREE_OF_PARALLELISM` for the `TPURGENT` consumer group.

Example

```
SELECT * FROM CS_RESOURCE_MANAGER.LIST_DEFAULT_RULES();
CONSUMER_GROUP ELAPSED_TIME_LIMIT IO_MEGABYTES_LIMIT SHARES CONCURRENCY_LIMIT
DEGREE_OF_PARALLELISM
-------------- ------------------ ------------------ ------ -----------------
--------------------
HIGH                            0                  0      4
3                     1
MEDIUM                          0                  0      2
1                     1
LOW                             0                  0      1
300                     1
TP                              0                  0      8
300                     1
TPURGENT                        0                  0     12
300
```

REVERT_TO_DEFAULT_VALUES Procedure

This procedure reverts the specified resource manager's plan properties to default values.

Syntax

```
CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(
   consumer_group         IN VARCHAR2,
   shares                 IN BOOLEAN   DEFAULT FALSE,
   concurrency_limit      IN BOOLEAN   DEFAULT FALSE);
```

Parameters

Parameter	Description
consumer_group	Specifies the consumer group to revert. Depending on the workload, valid values are: HIGH, MEDIUM, LOW, TP, or TPURGENT.
shares	When the value is TRUE, revert shares for the service to the default value.
concurrency_limit	When the value is TRUE, revert the concurrency_limit for the service to the default value. When you revert the concurrency_limit, both the concurrency_limit and the degree_of_parallelism values are set to their default values.

Usage Note

- When the workload type is Data Warehouse, the valid values for consumer_group are HIGH, MEDIUM, or LOW.

Examples

```
BEGIN
   CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(
      consumer_group => 'MEDIUM',
      concurrency_limit => TRUE);
END;
/

BEGIN
   CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(
      consumer_group => 'HIGH',
      shares => TRUE);
END;
/
```

UPDATE_PLAN_DIRECTIVE Procedure

Use this procedure to update the resource plan for a specified consumer group.

Syntax

```
CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(
   consumer_group         IN VARCHAR2,
   io_megabytes_limit     IN NUMBER   DEFAULT NULL,
   elapsed_time_limit     IN NUMBER   DEFAULT NULL,
   shares                 IN NUMBER   DEFAULT NULL,
   concurrency_limit      IN NUMBER   DEFAULT NULL);
```

Parameters

Parameter	Description
consumer_group	Specifies the consumer group to update. Depending on the workload, valid values are: HIGH, MEDIUM, LOW, TP, or TPURGENT.
io_megabytes_limit	Specifies the maximum megabytes of I/O that a SQL operation can issue. Specify a NULL value to clear the limit.
elapsed_time_limit	Specifies the maximum time in seconds that a SQL operation can run. Specify a NULL value to clear the limit.
shares	Specifies the shares value. A higher number of shares, relative to other consumer groups, increases the consumer group's CPU and I/O priority.
concurrency_limit	Specifies the maximum number of concurrent SQL statements that can be executed. This parameter is only valid with the MEDIUM consumer group.

Usage Notes

- When a SQL statement in the specified service runs more than the specified runtime limit (elapsed_time_limit) or does more I/O than the specified amount (io_megabytes_limit), then the SQL statement will be terminated.
- When the workload type is Data Warehouse, the valid values for consumer_group are HIGH, MEDIUM, or LOW.
- When the concurrency_limit parameter is specified, the only valid value for consumer_group is MEDIUM.

Examples

```
BEGIN
   CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(
      consumer_group => 'HIGH',
      shares => 8);
   CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(
      consumer_group => 'MEDIUM',
      shares => 2);
   CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(
      consumer_group => 'LOW',
      shares => 1);
END;
/

BEGIN
   CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(
      consumer_group => 'HIGH',
      io_megabytes_limit => null,
      elapsed_time_limit => null);
END;
/

BEGIN
```

```
   CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(
     consumer_group => 'MEDIUM',
     concurrency_limit => 2);
END;
/
```

CS_SESSION Package

The `CS_SESSION` package provides an interface to switch the database service and consumer group of the existing session.

When a connection is established with an Autonomous Database, that session is assigned a consumer group. For example, a session could be created using a connection to the LOW service of an Autonomous Database. You might want to switch the consumer group, for example from LOW to HIGH. The `CS_SESSION` package provides an API for switching.

- See Database Service Names for Autonomous Data Warehouse for more information.
- See Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database for more information.

The consumer groups determine concurrency and degree of parallelism (DOP). For example, statements on a connection established to the LOW database service run serially. Statements on a connection established to the HIGH database service run in parallel. If you have a workload that requires serial statement processing with switching to a HIGH consumer group for a few statements, the `CS_SESSION` package enables you to switch.

- SWITCH_SERVICE Procedure
  This procedure switches the database service and consumer group of the current session.

SWITCH_SERVICE Procedure

This procedure switches the database service and consumer group of the current session.

Syntax

```
CS_SESSION.SWITCH_SERVICE(service_name IN varchar2);
```

Parameters

Parameter	Description
`service_name`	Specifies the consumer group to update. Depending on the workload, valid values are: `HIGH`, `MEDIUM`, `LOW`, `TP`, or `TPURGENT`.

Usage Notes

When called, the procedure switches the session to the specified service and the related consumer group. If the specified service does not exist in that database, an error message is provided. For example, if you specify 'TP' as the service name on a data warehouse workload, the error indicates it's not a valid service name. No error is reported if the current service and the specified service are identical.

The procedure does not reset session attributes. Anything the user set for their session before calling this procedure will continue as-is. For example, if a session parameter was modified

and then later the session switched to a different service, the parameter value will stay the same.

Example

```
BEGIN
  CS_SESSION.SWITCH_SERVICE('HIGH');
END;
/
```

Security and Access

The `ADMIN` user is granted `EXECUTE` privilege on `CS_SESSION` with `GRANT OPTION`. The privilege is also granted to `DWROLE` without the `GRANT OPTION`.

Additional Security Considerations

If a user is granted `EXECUTE` privileges on this procedure and you do not want that user to switch to a specific service, you can use a `AFTER SET CONTAINER` trigger to block the operation. This is achieved by creating an `AFTER SET CONTAINER` trigger.

```
CREATE OR REPLACE TRIGGER SESS_SWITCH
AFTER SET CONTAINER ON DATABASE
BEGIN
IF SYS_CONTEXT('USERENV','SESSION_USER') = 'USER' and
   SYS_CONTEXT('USERENV','SERVICE_NAME') =
'serviceexample_low.adwc.oraclecloud.com'
THEN
    NULL;
ELSE
    RAISE_APPLICATION_ERROR(-20001, 'Denied!  You are not allowed to switch
service in the database');
END IF;
END;
/
```

Error Messages

The following table describes exceptions for `CS_SESSION`

Error	Message	Cause
20001	Invalid service name. Valid values are HIGH, MEDIUM, LOW.	For a data warehouse workload, a value other than 'HIGH', 'MEDIUM', 'LOW' was specified.
20001	Invalid service name. Valid values are HIGH, MEDIUM, LOW, TP, TPURGENT.	For a transaction processing workload, a value other than 'HIGH', 'MEDIUM', 'LOW', 'TP', 'TPURGENT' was specified.
20002	Service switch failed.	Failed to switch to the new service.

DBMS_SHARE Package

The `DBMS_SHARE` subprograms and views are used to share local data with external systems.

- Summary of Share Producer Subprograms
  This table lists the DBMS_SHARE package procedures and functions used to produce shares for recipients.
- Summary of Share Consumer Subprograms
  This table lists the DBMS_SHARE package procedures and functions used to consume shares.
- Summary of Share Producer Views
  This table lists the share producer views for the DBMS_SHARE package.
- Summary of Share Consumer Views
  This table lists the DBMS_SHARE package views.
- DBMS_SHARE Constants
  These constants are used by the DBMS_SHARE package.

Summary of Share Producer Subprograms

This table lists the DBMS_SHARE package procedures and functions used to produce shares for recipients.

Subprogram	Description
ADD_TO_SHARE Procedure	Add a table or view to a share.
ASSERT_SHAREABLE_OBJECT Procedure	Return without error, if the object exists and can be shared.
ASSERT_SHARING_ID Procedure	Run basic validation checks against a sharing id and return one in canonical form.
CAN_CREATE_SHARE Function	This function checks to see if the current schema can create share recipients.
CAN_CREATE_SHARE_RECIPIENT Function	This function checks to see if the current schema can create share recipients.
CLEAR_RECIPIENT_EVENTS Procedure	Clear events from the share recipient event log.
CLEAR_SHARE_EVENTS Procedure	Clear events from the share event log.
CREATE_BEARER_TOKEN_CREDENTIAL Procedure	Create a credential suitable for use with delta share providers.
CREATE_CLOUD_STORAGE_LINK Procedure	Create a named cloud storage URI link.
CREATE_OR_REPLACE_CLOUD_STORAGE_LINK Procedure	Create or replace a named cloud storage URI.
CREATE_OR_REPLACE_SHARE_RECIPIENT Procedure	Create or replace a share recipient.
CREATE_SHARE Procedure	Create a named share object.
CREATE_SHARE_RECIPIENT Procedure	Create a new share recipient.
DROP_CLOUD_STORAGE_LINK Procedure	Drop a cloud storage link.
DROP_RECIPIENT Procedure	Drop a recipient.
DROP_SHARE Procedure	Drop a share and all of its contents.

Subprogram	Description
DROP_SHARE_LINK_VIEW Procedure	Drop a view that was created by the `CREATE_SHARE_LINK_VIEW` procedure.
DROP_SHARE_VERSION Procedure	Drop a single share version.
DROP_SHARE_VERSIONS Procedure	Drop a range of share versions.
DROP_UNUSED_SHARE_VERSIONS Procedure	Drop any share version that is not currently in use.
ENABLE_SCHEMA Procedure	Enable or disable a schema for sharing.
GET_ACTIVATION_LINK Function	Generate the link that gets put into emails to the authorized recipient.
GET_PUBLISHED_IDENTITY Function	Get data about the current user that was set by `SET_PUBLISHED_IDENTITY`.
GET_RECIPIENT_PROPERTY Function	Return the value of a property for a recipient.
GET_SHARE_PROPERTY Function	Get the property value of an existing share.
GET_SHARE_TABLE_PROPERTY Function	Get the property value of an existing share table.
GRANT_TO_RECIPIENT Procedure	Grant access on a share to a specific recipient.
POPULATE_SHARE_PROFILE Procedure	Generate a delta profile for a recipient.
PUBLISH_SHARE Procedure	Publish a share and return immediately.
PUBLISH_SHARE_WAIT Procedure	Publish a share and wait until the background job is complete.
PURGE_DETACHED_FILES Procedure	Delete or forget parquet files that have become detached from their shares.
REMOVE_FROM_SHARE Procedure	Remove a table or view from a share.
RENAME_RECIPIENT Procedure	Rename a recipient.
RENAME_SHARE Procedure	Rename a share.
RENAME_SHARE_LINK Procedure	Rename a registered share link.
RENAME_SHARE_SCHEMA Procedure	Rename a share schema.
RENAME_SHARE_TABLE Procedure	Rename a share table.
REVOKE_FROM_RECIPIENT Procedure	Revoke access on a share from a specific recipient.
SET_CURRENT_SHARE_VERSION Procedure	Change the current version of a share.
SET_PUBLISHED_IDENTITY Procedure	Set data about the current user that will be supplied to recipients of published ORACLE shares.
SET_RECIPIENT_LOG_LEVEL Procedure	Change the log level for an existing share recipient.
SET_SHARE_LOG_LEVEL Procedure	Change the log level for an existing share.

Subprogram	Description
SET_STORAGE_CREDENTIAL Procedure	Set the access credential name for the given storage.
STOP_JOB Procedure	Stop a running share job.
UNPUBLISH_SHARE Procedure	Unpublish a share.
UPDATE_DEFAULT_RECIPIENT_PROPERTY Procedure	Update the default recipient property values.
UPDATE_DEFAULT_SHARE_PROPERTY Procedure	Update the default share property values.
UPDATE_RECIPIENT_PROPERTY Procedure	Update a property of an existing recipient.
UPDATE_SHARE_JOB_PROPERTY Procedure	Modify properties of a running share job.
UPDATE_SHARE_PROPERTY Procedure	Update a property of an existing share.
UPDATE_SHARE_TABLE_PROPERTY Procedure	Update the property value of an existing share table.
VALIDATE_CREDENTIAL Function	Validate a credential name, converting it to canonical form first if required.
VALIDATE_SHARE_STORAGE Procedure	Check to see if the given storage is suitable for versioned shares.
WAIT_FOR_JOB Procedure	This procedure waits until the specified share job is complete.

- ADD_TO_SHARE Procedure
  Add a table or view to a share. The object becomes visible to any external user who has been granted access to the share.
- ASSERT_SHAREABLE_OBJECT Procedure
  Return without error, if the object exists and can be shared.
- ASSERT_SHARING_ID Procedure
  Run basic validation checks against a sharing id and return one in canonical form. An exception is raised if the id is obviously invalid.
- CAN_CREATE_SHARE Function
  This function checks to see if the current schema can create share recipients. If shares can be created, a 1 is returned and 0 otherwise.
- CAN_CREATE_SHARE_RECIPIENT Function
  This function checks to see if the current schema can create share recipients. If shares can be created a 1 is returned, and 0 otherwise.
- CLEAR_RECIPIENT_EVENTS Procedure
  Clear events from the share recipient event log.
- CLEAR_SHARE_EVENTS Procedure
  Clear events from the share event log.
- CREATE_BEARER_TOKEN_CREDENTIAL Procedure
  Create a credential suitable for use with delta share providers.
- CREATE_CLOUD_STORAGE_LINK Procedure
  Create a named cloud storage URI link. A cloud storage link is a named association between an OCI bucket URI, and a local credential name.

- CREATE_OR_REPLACE_CLOUD_STORAGE_LINK Procedure
  Create or replace a named cloud storage URI. A cloud storage link is a named association between an OCI bucket URI, and a local credential name.
- CREATE_OR_REPLACE_SHARE_RECIPIENT Procedure
  Create or replace a share recipient. You must provide at least an email address or sharing id.
- CREATE_SHARE Procedure
  Create a named share object.
- CREATE_SHARE_RECIPIENT Procedure
  Create a new share recipient.
- DROP_CLOUD_STORAGE_LINK Procedure
  Drop a cloud storage link.
- DROP_RECIPIENT Procedure
  Drop a recipient. All access to the recipient will be revoked.
- DROP_SHARE Procedure
  Drop a share and all of its contents. Future access to the share by consumers will end.
- DROP_SHARE_LINK_VIEW Procedure
  Drop a view that was created by the `CREATE_SHARE_LINK_VIEW` procedure.
- DROP_SHARE_VERSION Procedure
  Drop a single share version. Note that you cannot drop the current version.
- DROP_SHARE_VERSIONS Procedure
  Drop a range of share versions. Note that you cannot drop the current version using this procedure.
- DROP_UNUSED_SHARE_VERSIONS Procedure
  Drop any share version that is not currently in use.
- ENABLE_SCHEMA Procedure
  Enable or disable a schema for sharing. This procedure must be run by the ADMIN user.
- GET_ACTIVATION_LINK Function
  Generate the link that gets put into emails to the authorized recipient. This activation link leads to the download page, where the recipient clicks a button to get the delta profile.
- GET_PUBLISHED_IDENTITY Function
  Get data about the current user that was set by `SET_PUBLISHED_IDENTITY`.
- GET_RECIPIENT_PROPERTY Function
  Return the value of a property for a recipient.
- GET_SHARE_PROPERTY Function
  Get the property value of an existing share.
- GET_SHARE_TABLE_PROPERTY Function
  Get the property value of an existing share table.
- GRANT_TO_RECIPIENT Procedure
  Grant access on a share to a specific recipient. The share and recipient must both belong to the same schema.
- POPULATE_SHARE_PROFILE Procedure
  Generate a delta profile for a recipient. You could print this to the screen or upload it somewhere. For example, to an object bucket using `DBMS_CLOUD.EXPORT_DATA`.
- PUBLISH_SHARE Procedure
  Publish a share and return immediately.

- PUBLISH_SHARE_WAIT Procedure
  Publish a share and wait until the background job is complete. The publication continues even if the call is interrupted.
- PURGE_DETACHED_FILES Procedure
  Delete or forget parquet files that have become detached from their shares.
- REMOVE_FROM_SHARE Procedure
  Remove a table or view from a share.
- RENAME_RECIPIENT Procedure
  Rename a recipient. This procedure only changes the local name of the recipient. The external definition of the recipient, for example the name of the OAUTH user or sharing id, is not changed.
- RENAME_SHARE Procedure
  Rename a share. Care should be take with this procedure since the change effects any existing consumers whose access is based on the previous name. Consumers are not notified directly or updated.
- RENAME_SHARE_LINK Procedure
  Rename a registered share link.
- RENAME_SHARE_SCHEMA Procedure
  Rename a share schema.
- RENAME_SHARE_TABLE Procedure
  Rename a share table.
- REVOKE_FROM_RECIPIENT Procedure
  Revoke access on a share from a specific recipient.
- SET_CURRENT_SHARE_VERSION Procedure
  Change the current version of a share.
- SET_PUBLISHED_IDENTITY Procedure
  Set data about the current user that will be supplied to recipients of published ORACLE shares.
- SET_RECIPIENT_LOG_LEVEL Procedure
  Change the log level for an existing share recipient.
- SET_SHARE_LOG_LEVEL Procedure
  Change the log level for an existing share.
- SET_STORAGE_CREDENTIAL Procedure
  Set the credential name used by the current user when it attempts to access the given storage.
- STOP_JOB Procedure
  Attempt to stop a running share job. The procedure should return quickly, but it may take some time for the associated job to stop.
- UNPUBLISH_SHARE Procedure
  Unpublish a share.
- UPDATE_DEFAULT_RECIPIENT_PROPERTY Procedure
  Update the default recipient property values. This procedure requires the user to have admin privileges.
- UPDATE_DEFAULT_SHARE_PROPERTY Procedure
  Update the default share property values.
- UPDATE_RECIPIENT_PROPERTY Procedure
  Update a property of an existing recipient.

- UPDATE_SHARE_JOB_PROPERTY Procedure
  Modify properties of a running share job. The procedure should return quickly, but it may take some time for the changes to take effect.
- UPDATE_SHARE_PROPERTY Procedure
  Update a property of an existing share.
- UPDATE_SHARE_TABLE_PROPERTY Procedure
  Update the property value of an existing share table.
- VALIDATE_CREDENTIAL Function
  Validate a credential name, converting it to canonical form first if required.
- VALIDATE_SHARE_STORAGE Procedure
  Check to see if the given storage is suitable for versioned shares.
- WAIT_FOR_JOB Procedure
  This procedure waits until the specified share job is complete.

ADD_TO_SHARE Procedure

Add a table or view to a share. The object becomes visible to any external user who has been granted access to the share.

Syntax

```
PROCEDURE ADD_TO_SHARE
(
    share_name           IN VARCHAR2,
    table_name           IN VARCHAR2,
    owner                IN VARCHAR2 := NULL,
    share_table_name     IN VARCHAR2 := NULL,
    share_schema_name    IN VARCHAR2 := NULL,
    object_metadata      IN SYS.JSON_OBJECT_T := NULL,
    replace_existing     IN BOOLEAN := FALSE,
    share_owner          IN VARCHAR2 := NULL,
    auto_commit          IN BOOLEAN := FALSE
);
```

Parameters

Parameter	Description
share_name	The name of an existing share to which the object is granted.
table_name	The name of the entity to share (for example, a table or view name).
owner	The owner of the entity to share. The default is to current schema.
share_table_name	The externally visible name of the table. By default, this is the uppercased table_name.
share_schema_name	The externally visible schema where the table will be placed. By default this is the uppercased table owner. The schema is created automatically if it does not already exist.
object_metadata	Optional metadata to associate with the shared entity.
replace_existing	If TRUE, and this share_table_name already exists, the existing table_name is dropped from the share and replaced with this table_name. If FALSE, and this share_table_name already exists, an exception is raised indicating the share table is already used.

Parameter	Description
share_owner	The owner of the share.
auto_commit	If TRUE, this procedure call commits changes that are not visible externally until the commit takes place. The default value is FALSE, which means that the user must COMMIT after running this call in order to make the change visible.

ASSERT_SHAREABLE_OBJECT Procedure

Return without error, if the object exists and can be shared.

Syntax

```
PROCEDURE ASSERT_SHAREABLE_OBJECT
(
  object_name          IN VARCHAR2,
  object_owner         IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
object_name	The name of the object.
object_owner	The owner of the object. Defaults to current schema.

ASSERT_SHARING_ID Procedure

Run basic validation checks against a sharing id and return one in canonical form. An exception is raised if the id is obviously invalid.

Syntax

```
PROCEDURE ASSERT_SHARING_ID
(
    sharing_id              IN OUT NOCOPY VARCHAR2,
    out_sharing_id_type     IN OUT NOCOPY VARCHAR2
);
```

Parameters

Parameter	Description
sharing_id	The id to check.
out_sharing_id_type	The type of the id, if valid. For example, TENANCY or DATABASE.

CAN_CREATE_SHARE Function

This function checks to see if the current schema can create share recipients. If shares can be created, a 1 is returned and 0 otherwise.

Syntax

```
FUNCTION CAN_CREATE_SHARE
RETURN NUMBER;
```

Example: Before and after enabling the admin schema

```
SQL> select dbms_share.can_create_share from dual;
 CAN_CREATE_SHARE
----------------
               0
SQL> exec dbms_share.enable_schema('admin')
PL/SQL procedure successfully completed.
SQL> select dbms_share.can_create_share from dual;
CAN_CREATE_SHARE
----------------
               1
```

CAN_CREATE_SHARE_RECIPIENT Function

This function checks to see if the current schema can create share recipients. If shares can be created a 1 is returned, and 0 otherwise.

Syntax

```
FUNCTION CAN_CREATE_SHARE_RECIPIENT
RETURN NUMBER;
```

CLEAR_RECIPIENT_EVENTS Procedure

Clear events from the share recipient event log.

Syntax

```
PROCEDURE CLEAR_RECIPIENT_EVENTS
(
   recipient_name       IN VARCHAR2,
   from_time            IN TIMESTAMP WITH TIME ZONE := NULL,
   to_time              IN TIMESTAMP WITH TIME ZONE := NULL,
   recipient_owner      IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
recipient_name	The local name of the share recipient.

Parameter	Description
from_time	Earliest time for events that should be cleared or NULL.
to_time	Latest time for events that should be cleared or NULL.
recipient_owner	The schema that owns the recipient.

CLEAR_SHARE_EVENTS Procedure

Clear events from the share event log.

Syntax

```
PROCEDURE CLEAR_SHARE_EVENTS
(
    share_name           IN VARCHAR2,
    from_time            IN TIMESTAMP WITH TIME ZONE := NULL,
    to_time              IN TIMESTAMP WITH TIME ZONE := NULL,
    share_owner          IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
share_name	The name of the share.
from_time	Earliest time for events that should be cleared or NULL.
to_time	Latest time for events that should be cleared or NULL.
share_owner	The schema that owns the share.

CREATE_BEARER_TOKEN_CREDENTIAL Procedure

Create a credential suitable for use with delta share providers.

This is similar to the CREATE_CREDENTIALS call, but it takes explicit values instead of a delta sharing profile. See CREATE_CREDENTIALS Procedure and Function for further information.

Syntax

```
PROCEDURE CREATE_BEARER_TOKEN_CREDENTIAL
(
   credential_name      IN VARCHAR2,
   bearer_token         IN VARCHAR2 := NULL,
   token_endpoint       IN VARCHAR2 := NULL,
   client_id            IN VARCHAR2 := NULL,
   client_secret        IN VARCHAR2 := NULL,
   token_refresh_rate   IN PLS_INTEGER := 3600
);
```

Parameters

Parameter	Description
credential_name	The name of the new credential.

Parameter	Description
bearer_token	The bearer token, if known.
token_endpoint	The endpoint to call to get a new token.
client_id	The username to send to the token_endpoint.
client_secret	The password to send to the token_endpoint.
token_refresh_rate	An optional refresh time, in seconds.

Example: Credential with a fixed bearer token

In its simplest form, this procedure is equivalent to calling DBMS_CREDENTIAL with a user name of'BEARER_TOKEN' and the bearer token itself as password.

```
SQL> exec dbms_share.create_bearer_token_credential('MY_FIXED_CREDENTIAL',
'FF42322D27D4C2DEE05392644664351E')
PL/SQL procedure successfully completed.
SQL> select username from user_credentials where credential_name =
'MY_FIXED_CREDENTIAL';
USERNAME
--------------------------------------------------------------------------------
----------------------------------------
BEARER_TOKEN
```

Example: Credential with a renewable bearer token

Create a credential that contains a short lived bearer token obtained from a token endpoint. The bearer token will be refreshed once every hour using a second credential, which is populated from the client id and secret.

```
SQL> BEGIN
  2    dbms_share.create_bearer_token_credential(
  3      credential_name=>'MY_RENEWABLE_CREDENTIAL',
  4      token_endpoint=>'https://myserver/ords/share_provider/oauth/token',
  5      client_id=>'VXGQ_44s6qJ-K4WHUNM2yQ..',
  6      client_secret=>'y9ddppgwEmZl7adDHFQndw..');
  7  END;
  8  /
 PL/SQL procedure successfully completed.
 SQL> select credential_name, username from user_credentials where
credential_name LIKE 'MY_RENEWABLE_CREDENTIAL%';
 CREDENTIAL_NAME                             USERNAME
----------------------------------------
-----------------------------------
MY_RENEWABLE_CREDENTIAL                      BEARER_TOKEN
MY_RENEWABLE_CREDENTIAL$TOKEN_REFRESH_CRED   VXGQ_44s6qJ-K4WHUNM2yQ..
```

CREATE_CLOUD_STORAGE_LINK Procedure

Create a named cloud storage URI link. A cloud storage link is a named association between an OCI bucket URI, and a local credential name.

> **Note:**
>
> Use the SET_STORAGE_CREDENTIAL Procedure to add a credential to the storage link.

Syntax

```
PROCEDURE CREATE_CLOUD_STORAGE_LINK
(
  storage_link_name    IN VARCHAR2,
  uri                  IN VARCHAR2,
  owner                IN VARCHAR2 := NULL,
  metadata             IN SYS.JSON_OBJECT_T := NULL,
  auto_commit          IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
storage_link_name	The name of the cloud storage link. The name of the link should follow standard Oracle naming conventions.
uri	The URI of the storage bucket. The URI should be of the form `https://objectstorage.`*region*`.oraclecloud.com/n/`*namespace-string*`/b/`*bucket*`/o`
owner	The owner of the storage link. Leave as NULL for the current user.
metadata	(Optional) A JSON metadata document containing additional information.
auto_commit	The default is TRUE. If TRUE, this transaction is committed. If FALSE, the user must commit the transaction. Changes are not visible until the commit takes place.

Example

In this example, a cloud storage link named `MY_SHARE_STORAGE` is created on the given URL.

```
SQL> BEGIN
  2    dbms_share.create_cloud_storage_link(
  3      'MY_SHARE_STORAGE',
  4      'https://objectstorage.../n/abcdef/b/my_bucket/o' );
  5  END;
  6  /

PL/SQL procedure successfully completed.

SQL> select storage_link_name from user_lineage_cloud_storage_links;

STORAGE_LINK_NAME
```

```
--------------------------------------------------------------------------------
----------
MY_SHARE_STORAGE
```

CREATE_OR_REPLACE_CLOUD_STORAGE_LINK Procedure

Create or replace a named cloud storage URI. A cloud storage link is a named association between an OCI bucket URI, and a local credential name.

> **Note:**
>
> Use the SET_STORAGE_CREDENTIAL Procedure to add a credential to the storage link.

Syntax

```
PROCEDURE CREATE_OR_REPLACE_CLOUD_STORAGE_LINK
(
    storage_link_name    IN VARCHAR2,
    uri                  IN VARCHAR2,
    owner                IN VARCHAR2 := NULL,
    metadata             IN SYS.JSON_OBJECT_T := NULL,
    auto_commit          IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
storage_link_name	The name of the cloud storage link. The name of the link should follow standard Oracle naming conventions. See Database Object Names and Qualifiers
uri	The URI of the storage bucket. The URI should be of the form *https://objectstorage.region.oraclecloud.com/n/namespace-string/b/bucket/o*
owner	The owner of the storage link. Defaults to the current schema.
metadata	An optional JSON metadata document containing additional information.
auto_commit	If TRUE, the changes automatically commit after creating the link. The default is TRUE.

CREATE_OR_REPLACE_SHARE_RECIPIENT Procedure

Create or replace a share recipient. You must provide at least an email address or sharing id.

Syntax

```
PROCEDURE CREATE_OR_REPLACE_SHARE_RECIPIENT
(
    recipient_name       IN VARCHAR2,
    description          IN VARCHAR2 := NULL,
    recipient_owner      IN VARCHAR2 := NULL,
    email                IN VARCHAR2 := NULL,
```

```
    sharing_id           IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
recipient_name	The local name of the share recipient. Some names are not allowed (e.g. MY_TENANCY).
description	A description of the recipient.
recipient_owner	The schema that owns the recipient.
email	An email that will be registered for the OAUTH user.
sharing_id	The sharing id of the recipient, from GET_SHARING_ID Function.

CREATE_SHARE Procedure

Create a named share object.

Syntax

```
PROCEDURE CREATE_SHARE
(
  share_name           IN VARCHAR2,
  share_type           IN VARCHAR2 := SHARE_TYPE_VERSIONED,
  storage_link_name    IN VARCHAR2 := NULL,
  storage_link_owner   IN VARCHAR2 := NULL,
  description          IN VARCHAR2 := NULL,
  public_description   IN VARCHAR2 := NULL,
  configuration        IN SYS.JSON_OBJECT_T := NULL,
  force_create         IN BOOLEAN  := FALSE,
  share_owner          IN VARCHAR2 := NULL,
  auto_commit          IN BOOLEAN := FALSE,
  log_level            IN PLS_INTEGER := LOG_LEVEL_BASIC,
  run_storage_tests    IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
share_name	The name of the share. This name is uppercased since delta shares are case insensitive. The name follows standard Oracle conventions, so it must be 128 characters or fewer and must be double quoted if it is not a simple identifier. The only difference is that it will be uppercased even if it is double quoted.
share_type	The type of share. For information on constants used for this parameter, see descriptions for **Share Types** in DBMS_SHARE Constants.
storage_link_name	The name of the cloud storage link where the objects are created. The user must have read/write access to this storage and have the ability to create pre-authenticated requests on the storage. The parameter is required for versioned shares, and optional for current shares.
storage_link_owner	The owner of the cloud storage link where objects are created.
description	A local description for the share.
public_description	A public description for the share.

Parameter	Description
configuration	A configuration document that defines how objects are created.
force_create	Set force_create to TRUE to redefine the share if it exists.
share_owner	The owner of the share.
auto_commit	If TRUE, this procedure call commits changes that are not visible externally until the commit takes place. The default value is FALSE, which means that the user must COMMIT after running this call in order to make the change visible.
log_level	Event logging level. This controls the amount of detail recorded in the ALL_SHARE_EVENTS View. For information on constants used for this parameter, see descriptions for **Log Level** in DBMS_SHARE Constants.
run_storage_tests	If this parameter is TRUE then DBMS_SHARE runs tests to verify that the specified share storage link has the correct privileges. If this parameter is FALSE, then the procedure does not run any checks at creation time. This may lead to errors later, during publication or consumption of the share. Oracle recommends that you specify TRUE for this parameter.

CREATE_SHARE_RECIPIENT Procedure

Create a new share recipient.

Syntax

```
PROCEDURE CREATE_SHARE_RECIPIENT
(
    recipient_name       IN VARCHAR2,
    description          IN VARCHAR2 := NULL,
    recipient_owner      IN VARCHAR2 := NULL,
    email                IN VARCHAR2 := NULL,
    sharing_id           IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
recipient_name	The local name of the share recipient. Some names are not allowed, for example: MY_TENANCY.
description	A description of the recipient.
recipient_owner	The schema that owns the recipient.
email	An email that will be registered for the OAUTH user. You must provide at least one of email or sharing id.
sharing_id	The sharing id of the recipient from GET_SHARING_ID Function. You must provide at least one of email or sharing id.

DROP_CLOUD_STORAGE_LINK Procedure

Drop a cloud storage link.

Syntax

```
PROCEDURE DROP_CLOUD_STORAGE_LINK
(
    storage_link_name    IN VARCHAR2,
    owner                IN VARCHAR2 := NULL,
    auto_commit          IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
storage_link_name	The name of the cloud storage link to drop.
owner	The owner of the cloud storage link. Leave as NULL for the current user.
auto_commit	If TRUE, the code automatically commits after dropping the link. The default is TRUE.

DROP_RECIPIENT Procedure

Drop a recipient. All access to the recipient will be revoked.

Syntax

```
PROCEDURE DROP_RECIPIENT
(
    recipient_name       IN VARCHAR2,
    owner                IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
recipient_name	The name of the share recipient.
owner	The schema that defines the recipient.

DROP_SHARE Procedure

Drop a share and all of its contents. Future access to the share by consumers will end.

Syntax

```
PROCEDURE DROP_SHARE
(
  share_name           IN VARCHAR2,
  share_owner          IN VARCHAR2 := NULL,
```

```
  destroy_objects       IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
share_name	The name of the share to drop.
share_owner	The owner of the share to drop.
destroy_objects	If TRUE, delete all objects created on behalf of the share. The default is TRUE.

DROP_SHARE_LINK_VIEW Procedure

Drop a view that was created by the `CREATE_SHARE_LINK_VIEW` procedure.

See CREATE_SHARE_LINK_VIEW Procedure for further information.

Syntax

```
PROCEDURE DROP_SHARE_LINK_VIEW
(
    view_name             IN VARCHAR2,
    view_owner            IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
view_name	The name of the new view.
view_owner	The view owner. The default is the current schema.

DROP_SHARE_VERSION Procedure

Drop a single share version. Note that you cannot drop the current version.

Syntax

```
PROCEDURE DROP_SHARE_VERSION
(
    share_name            IN VARCHAR2,
    share_version         IN NUMBER,
    destroy_objects       IN BOOLEAN := TRUE,
    force_drop            IN BOOLEAN := FALSE,
    share_owner           IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
share_name	The name of the share.

Parameter	Description
share_version	The version to drop. You can not drop the current version.
destroy_objects	Destroy any associated object in cloud storage, if applicable.
force_drop	Drop the share version even if there is an outstanding PAR file on the version.
share_owner	The owner of the share.

DROP_SHARE_VERSIONS Procedure

Drop a range of share versions. Note that you cannot drop the current version using this procedure.

Syntax

```
PROCEDURE DROP_SHARE_VERSIONS
(
    share_name           IN VARCHAR2,
    from_share_version   IN NUMBER,
    to_share_version     IN NUMBER,
    destroy_objects      IN BOOLEAN := TRUE,
    force_drop           IN BOOLEAN := FALSE,
    share_owner          IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
share_name	The name of the share.
from_share_version	The lowest version to drop.
to_share_version	The highest version to drop.
destroy_objects	Destroy any associated object in cloud storage, if applicable.
force_drop	Drop the share version even if there is an outstanding PAR file on the version.
share_owner	The owner of the share.

DROP_UNUSED_SHARE_VERSIONS Procedure

Drop any share version that is not currently in use.

Syntax

```
PROCEDURE DROP_UNUSED_SHARE_VERSIONS
(
    share_name           IN VARCHAR2,
    destroy_objects      IN BOOLEAN := TRUE,
    share_owner          IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
share_name	The name of the share.
destroy_objects	Destroy any associated object in cloud storage, if applicable.
share_owner	The owner of the share.

ENABLE_SCHEMA Procedure

Enable or disable a schema for sharing. This procedure must be run by the ADMIN user.

Users can consume delta shares without being enabled with this procedure, but they cannot create or publish shares. Sharing is disabled by default for all schemas, including ADMIN.

Syntax

```
PROCEDURE ENABLE_SCHEMA
(
    schema_name           IN VARCHAR2,
    enabled               IN BOOLEAN := TRUE,
    privileges            IN PLS_INTEGER := NULL
);
```

Parameters

Parameter	Description
schema_name	The schema to enable or disable.
enable	TRUE to enable, FALSE to disable.

Parameter	Description
privileges	The privileges argument has bitmap values. If you leave the argument as NULL, it defaults to PRIV_CREATE_SHARE + PRIV_CREATE_RECIPIENT + PRIV_CONSUME_ORACLE_SHARE. Bitmap values are as follows: • PRIV_CREATE_SHARE Allow users to create an publish shares in their own schema. PRIV_CREATE_SHARE CONSTANT PLS_INTEGER := 1; • PRIV_CREATE_RECIPIENT Allow users to create share recipients in their own schema. PRIV_CREATE_RECIPIENT CONSTANT PLS_INTEGER := 2; • PRIV_CONSUME_ORACLE_SHARE Allow users to consume Oracle-to-Oracle live shares. PRIV_CONSUME_ORACLE_SHARE CONSTANT PLS_INTEGER := 4; • PRIV_ORDS_ACL Grant an ACL to the user on the local ORDS endpoint. This is required for the user to generate bearer tokens on locally created shares. PRIV_ORDS_ACL CONSTANT PLS_INTEGER := 8;

GET_ACTIVATION_LINK Function

Generate the link that gets put into emails to the authorized recipient. This activation link leads to the download page, where the recipient clicks a button to get the delta profile.

Syntax

```
FUNCTION GET_ACTIVATION_LINK
(
  recipient_name       IN VARCHAR2,
  recipient_owner      IN VARCHAR2 := NULL,
  expiration           IN PLS_INTEGER := 259200,
  invalidate_previous  IN BOOLEAN := TRUE
)
RETURN VARCHAR2;
```

Parameters

Parameter	Description
recipient_name	The local name of the recipient.
recipient_owner	The schema that owns the recipient.
expiration	Number of seconds before the activation link expires.
invalidate_previous	If TRUE, which is the default, a previously generated activation link is made invalid. If FALSE, a previously generated activation link remains valid.

Example: Print activation link to the screen

```
SQL> exec dbms_share.create_share_recipient('new_recipient',
email=>'anyone@example.com')

PL/SQL procedure successfully completed.

SQL> column PROFILE format A200
SQL> variable sprof varchar2(32767)
SQL> declare
  2    profile sys.json_object_t;
  3  begin
  4    dbms_share.populate_share_profile('NEW_RECIPIENT', profile);
  5    :sprof := profile.to_string;
  6  end;
  7  /
PL/SQL procedure successfully completed.

SQL> exec
dbms_output.put_line(dbms_share.get_activation_link('NEW_RECIPIENT'))
http://.../ords/_adpshr/delta-sharing/download?key=43BA....YXJlX3Rlc3Q=

PL/SQL procedure successfully completed.
```

GET_PUBLISHED_IDENTITY Function

Get data about the current user that was set by `SET_PUBLISHED_IDENTITY`.

Syntax

```
FUNCTION GET_PUBLISHED_IDENTITY
  RETURN CLOB;
```

Example

```
SQL> declare
  2    id_json json_object_t := json_object_t();
  3  begin
  4    id_json.put('name', 'Demo Publisher');
  5    id_json.put('description', 'Documentation Share Provider');
  6    id_json.put('contact', 'null@example.com');
  7    dbms_share.set_published_identity(id_json);
  8  end;
  9  /

PL/SQL procedure successfully completed.

SQL> select json_query(dbms_share.get_published_identity, '$' pretty)
"Published Identity"
  2  from dual;

Published Identity-
--------------------------------------------------------------------------------
```

```
-
{
  "name" : "Demo Publisher",
  "description" : "Documentation Share Provider",
  "contact" : "null@example.com"
}
```

GET_RECIPIENT_PROPERTY Function

Return the value of a property for a recipient.

Syntax

```
FUNCTION GET_RECIPIENT_PROPERTY
(
  recipient_name       IN VARCHAR2,
  recipient_property   IN VARCHAR2,
  recipient_owner      IN VARCHAR2 := NULL
)RETURN VARCHAR2;
```

Parameters

Parameter	Description
recipient_name	The name of the recipient.
recipient_property	The property to get. These properties include: • PROP_SHARE_DESC • PROP_SHARE_PUBLIC_DESC • PROP_SHARE_VERSION_ACCESS • PROP_SHARE_JOB_NAME • PROP_SHARE_SPLIT_SIZE • PROP_SHARE_LOG_LEVEL • PROP_SHARE_JOB_DOP • PROP_SHARE_JOB_CLASS • PROP_SHARE_JOB_PRIORITY • PROP_SHARE_VERSION_ACCESS For information on constants used for this parameter, see descriptions for **Share Properties** in DBMS_SHARE Constants.
recipient_owner	The owner of the recipient. Defaults to current user.

GET_SHARE_PROPERTY Function

Get the property value of an existing share.

Syntax

```
FUNCTION GET_SHARE_PROPERTY
(
  share_name       IN VARCHAR2,
  share_property   IN VARCHAR2,
  share_owner      IN VARCHAR2 := NULL
)
RETURN VARCHAR2
```

Parameters

Parameter	Description
share_name	The name of the share.
share_property	The property value to get. For information on constants used for this parameter, see descriptions for **Share Properties** in DBMS_SHARE Constants.
share_owner	The owner of the share. Defaults to current user.

GET_SHARE_TABLE_PROPERTY Function

Get the property value of an existing share table.

Syntax

```
FUNCTION GET_SHARE_TABLE_PROPERTY
(
  share_name           IN VARCHAR2,
  share_table_name     IN VARCHAR2,
  share_table_property IN VARCHAR2,
  share_schema_name    IN VARCHAR2 := NULL,
  share_owner          IN VARCHAR2 := NULL
)RETURN VARCHAR2;
```

Parameters

Parameter	Description
share_name	The name of the share.
share_table_name	The name of the share table.
share_table_property	The share table property to update. For information on constants used for this parameter, see descriptions for **Share Table Properties** in DBMS_SHARE Constants.
share_schema_name	The name of the share schema. Defaults to uppercase of current user.
share_owner	The owner of the share.

GRANT_TO_RECIPIENT Procedure

Grant access on a share to a specific recipient. The share and recipient must both belong to the same schema.

Syntax

```
PROCEDURE GRANT_TO_RECIPIENT
(
    share_name           IN VARCHAR2,
    recipient_name       IN VARCHAR2,
    owner                IN VARCHAR2 := NULL,
    auto_commit          IN BOOLEAN := FALSE
);
```

Parameters

Parameter	Description
share_name	The name of the share to grant.
recipient_name	The name of the recipient.
owner	The owner of both the share and recipient.
auto_commit	The auto_commit parameter is ignored. This procedure will always commit.

POPULATE_SHARE_PROFILE Procedure

Generate a delta profile for a recipient. You could print this to the screen or upload it somewhere. For example, to an object bucket using DBMS_CLOUD.EXPORT_DATA.

Syntax

```
PROCEDURE POPULATE_SHARE_PROFILE
(
    recipient_name       IN VARCHAR2,
    share_profile        IN OUT NOCOPY SYS.JSON_OBJECT_T,
    recipient_owner      IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
recipient_name	The local name of the recipient.
share_profile	The share profile, without bearer token.
recipient_owner	The schema that owns the recipient.

Example: Print the profile to the screen.

```
SQL> EXEC DBMS_SHARE.CREATE_SHARE_RECIPIENT('new_recipient',
email=>'anyone@example.com')

PL/SQL procedure successfully completed.
SQL> column PROFILE format A200
SQL> variable sprof varchar2(32767)
SQL> declare
  2    profile sys.json_object_t;
  3  begin
  4    dbms_share.populate_share_profile('NEW_RECIPIENT', profile);
  5    :sprof := profile.to_string;
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL> select json_query(:sprof, '$' pretty) "PROFILE" from dual;
PROFILE--------------------------------------------------------------------------
-----------------------------------------------
```

```
{
  "shareCredentialsVersion" : 1,
  "endpoint" : "https://.../ords/share_test/_delta_sharing/",
  "bearerToken" : "mc7puvhqCpU6xjTOjRdl_w",
  "tokenEndpoint" : "https://.../ords/share_test/oauth/token",
  "clientID" : "VXGQ_44s6qJ-K4WHUNM2yQ..",
  "clientSecret" : "y9ddppgwEmZl7adDHFQndw.."
}
```

PUBLISH_SHARE Procedure

Publish a share and return immediately.

The publication will continue in the background. You can check on the status of the job by querying the `USER_SHARE_JOBS` view. See USER_SHARE_JOBS View for further information.

Syntax

```
PROCEDURE PUBLISH_SHARE
(
    share_name           IN VARCHAR2,
    share_owner          IN VARCHAR2 := NULL,
    drop_prior_versions  IN BOOLEAN := FALSE,
    share_job_dop        IN NUMBER := NULL,
    share_job_priority   IN NUMBER := NULL,
    job_class            IN VARCHAR2 := 'DEFAULT_JOB_CLASS',
    force_job_class      IN BOOLEAN := FALSE
);
```

Parameters

Parameter	Description
`share_name`	The name of the share to publish.
`share_owner`	The share owner, which must be the current user or NULL.
`drop_prior_versions`	Set to TRUE if you want to drop all prior versions of the share. Note that versions are only dropped if there are no outstanding Pre-Authenticated Requests (PARs).
`share_job_dop`	Specify the maximum number of `dbms_scheduler` jobs that will be used to publish the share. Leave as NULL to use the system default number.
`share_job_priority`	Specify a relative priority of this share publication compared to others. If two shares are being published at the same time, then the one with the highest priority is processed first even if it was started later.
`job_class`	The scheduler job class, from `all_scheduler_job_classes`, that are used to publish the share.
`force_job_class`	Use the specified`job_class` even if the admin has defined a different default job class.

PUBLISH_SHARE_WAIT Procedure

Publish a share and wait until the background job is complete. The publication continues even if the call is interrupted.

Syntax

```
PROCEDURE PUBLISH_SHARE_WAIT
(
       share_name           IN VARCHAR2,
       share_owner          IN VARCHAR2 := NULL,
       drop_prior_versions  IN BOOLEAN := FALSE,
       share_job_dop        IN NUMBER := NULL,
       share_job_priority   IN NUMBER := NULL,
       job_class            IN VARCHAR2 := 'DEFAULT_JOB_CLASS',
       force_job_class      IN BOOLEAN := FALSE
);
```

Parameters

Parameter	Description
share_name	The name of the share to publish.
share_owner	The share owner, which must be the current user or NULL.
drop_prior_versions	Set to TRUE if you want to drop all prior versions of the share. Note that versions are only dropped if there are no outstanding Pre-Authenticated Requests (PARs).
share_job_dop	Specify the maximum number of dbms_scheduler jobs that will be used to publish the share. Leave as NULL to use the system default number.
share_job_priority	Specify a relative priority of this share publication compared to others. If two shares are being published at the same time, then the one with the highest priority is processed first even if it was started later.
job_class	The scheduler job class, from all_scheduler_job_classes, that are used to publish the share.
force_job_class	Use the specifiedjob_class even if the admin has defined a different default job class.

PURGE_DETACHED_FILES Procedure

Delete or forget parquet files that have become detached from their shares.

Syntax

```
PROCEDURE PURGE_DETACHED_FILES
(
    file_pattern         IN VARCHAR2 := '%',
    credential_name      IN VARCHAR2 := NULL,
    purge_mode           IN PLS_INTEGER := PURGE_DROP,
    owner_id             IN NUMBER := SYS_CONTEXT('USERENV',
'CURRENT_USERID')
);
```

A version of PURGE_DETACHED_FILES that returns the ID of the purge job, if any.

```
PROCEDURE PURGE_DETACHED_FILES
(
    purge_job_id         IN OUT NOCOPY NUMBER,
    file_pattern         IN VARCHAR2 := '%',
```

```
    credential_name      IN VARCHAR2 := NULL,
    purge_mode           IN PLS_INTEGER := PURGE_DROP,
    owner_id             IN NUMBER := SYS_CONTEXT('USERENV',
'CURRENT_USERID')
);
```

Parameters

Parameter	Description
purge_job_id	The purge job ID.
file_pattern	An optional LIKE pattern for the files that are to be purged.
credential_name	An optional credential to be used to delete the files.
purge_mode	Specifies how the files are purged. Purge mode values include: • PURGE_DROP Attempt to drop the files, using the specified credential. If the file cannot be dropped, then the file will continue to be listed in the *_SHARE_DETACHED_FILES views. PURGE_DROP CONSTANT PLS_INTEGER := 1; • PURGE_DROP_FORCE Attempt to drop the files, using the specified credential. The file will be removed from the *_SHARE_DETACHED_FILES views even if the attempt to drop the file fails again. PURGE_DROP_FORCE CONSTANT PLS_INTEGER := 2; • PURGE_FORGET Remove the files *_SHARE_DETACHED_FILES views without attempting to drop them. PURGE_FORGET CONSTANT PLS_INTEGER := 3;
owner_id	The owner ID whose files are to be purged.

REMOVE_FROM_SHARE Procedure

Remove a table or view from a share.

Syntax

```
PROCEDURE REMOVE_FROM_SHARE
(
    share_name           IN VARCHAR2,
    share_table_name     IN VARCHAR2,
    share_schema_name    IN VARCHAR2 := NULL,
    share_owner          IN VARCHAR2 := NULL,
    auto_commit          IN BOOLEAN := FALSE
);
```

Parameters

Parameter	Description
share_name	The name of an existing share to which the object is revoked.
share_table_name	The name of the share table to revoke. This must match the externally visible name, so the input will be uppercased.
share_schema_name	The name of the share schema. Defaults to uppercase of current user.
share_owner	The owner of the share.
auto_commit	If TRUE, this procedure call commits changes that are not visible externally until the commit takes place. The default value is FALSE, which means that the user must COMMIT after running this call in order to make the change visible.

RENAME_RECIPIENT Procedure

Rename a recipient. This procedure only changes the local name of the recipient. The external definition of the recipient, for example the name of the OAUTH user or sharing id, is not changed.

Syntax

```
PROCEDURE RENAME_RECIPIENT
(
    old_recipient_name   IN VARCHAR2,
    new_recipient_name   IN VARCHAR2,
    owner                IN VARCHAR2 := NULL,
    auto_commit          IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
old_recipient_name	The current name of the share recipient.
new_recipient_name	The new name of the share recipient.
owner	The schema that defines the recipient.
auto_commit	If TRUE, the changes are automatically committed. Changes are not visible externally until the commit takes place. The default is TRUE.

RENAME_SHARE Procedure

Rename a share. Care should be take with this procedure since the change effects any existing consumers whose access is based on the previous name. Consumers are not notified directly or updated.

Syntax

```
PROCEDURE RENAME_SHARE
(
    old_share_name       IN VARCHAR2,
    new_share_name       IN VARCHAR2,
    share_owner          IN VARCHAR2 := NULL,
```

```
    auto_commit           IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
old_share_name	The current name of the share.
new_share_name	The new name of the share.
share_owner	The owner of the share.
auto_commit	If TRUE, the changes are automatically committed. Changes are not visible externally until the commit takes place. The default is TRUE.

RENAME_SHARE_LINK Procedure

Rename a registered share link.

Syntax

```
PROCEDURE RENAME_SHARE_LINK
(
    old_name              IN VARCHAR2,
    new_name              IN VARCHAR2,
    link_owner            IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
old_name	The current name of the share link.
new_name	The new name of the link.
link_owner	The owner of the share link. Defaults to current schema.

RENAME_SHARE_SCHEMA Procedure

Rename a share schema.

Syntax

```
PROCEDURE RENAME_SHARE_SCHEMA
(
    share_name            IN VARCHAR2,
    old_schema_name       IN VARCHAR2,
    new_schema_name       IN VARCHAR2,
    share_owner           IN VARCHAR2 := NULL,
    auto_commit           IN BOOLEAN := FALSE
);
```

Parameters

Parameter	Description
share_name	The name of the share.
old_schema_name	The old name of the schema.
new_schema_name	The new name of the schema.
share_owner	The owner of the share. Defaults to the current schema.
auto_commit	If TRUE, the changes are automatically committed. Changes are not visible externally until the commit takes place. The default is FALSE.

RENAME_SHARE_TABLE Procedure

Rename a share table.

Syntax

```
PROCEDURE RENAME_SHARE_TABLE
(
    share_name           IN VARCHAR2,
    share_schema_name    IN VARCHAR2,
    old_share_table_name IN VARCHAR2,
    new_share_table_name IN VARCHAR2,
    share_owner          IN VARCHAR2 := NULL,
    auto_commit          IN BOOLEAN := FALSE
);
```

Parameters

Parameter	Description
share_name	The name of the share.
share_schema_name	The name of the share schema.
old_share_table_name	The old name of the share table.
new_share_table_name	The new name of the share table.
share_owner	The owner of the share.
auto_commit	If TRUE, the changes are automatically committed. Changes are not visible externally until the commit takes place. The default is FALSE.

REVOKE_FROM_RECIPIENT Procedure

Revoke access on a share from a specific recipient.

Syntax

```
PROCEDURE REVOKE_FROM_RECIPIENT
(
    share_name           IN VARCHAR2,
    recipient_name       IN VARCHAR2,
    owner                IN VARCHAR2 := NULL,
    auto_commit          IN BOOLEAN := FALSE
);
```

Parameters

Parameter	Description
share_name	The name of the share to revoke.
recipient_name	The name of the recipient.
owner	The owner of the share and recipient.
auto_commit	If TRUE, the changes are automatically committed. Changes are not visible externally until the commit takes place. The default is FALSE.

SET_CURRENT_SHARE_VERSION Procedure

Change the current version of a share.

Syntax

```
PROCEDURE SET_CURRENT_SHARE_VERSION
(
    share_name          IN VARCHAR2,
    share_version       IN NUMBER,
    share_owner         IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
share_name	The name of the share.
share_version	The new version or NULL. The version must exist and must be valid. If the share_version is NULL, then no version will be marked as CURRENT and the share will be "unpublished".
share_owner	The owner of the share. Defaults to the current schema.

SET_PUBLISHED_IDENTITY Procedure

Set data about the current user that will be supplied to recipients of published ORACLE shares.

Syntax

```
PROCEDURE SET_PUBLISHED_IDENTITY
(
    metadata              IN SYS.JSON_OBJECT_T := NULL
);
```

Parameters

Parameter	Description
metadata	If the provider identity has already been set, then only the items that the caller wants to update need to be included in the JSON. Supplying a NULL value for an item causes that item to be removed from the stored identity. However, "name", "description" and "contact" cannot be removed in this way. If the metadata argument is NULL, the existing provider identity is deleted. This can only happen if the current user has no published shares. If the provider identity has not yet been set, the JSON must contain at least: • name (<=128 bytes) • description (<=4000 bytes) • contact (<=4000 bytes) values Additional items may be included at the caller's discretion.

Example: Include additional items in the JSON metadata

In addition to the required "name", "description", and "contact", this example includes additional item "schedule".

```
{
  "name" : "A sample share provider",
  "description" : "Test of share provider metadata",
  "contact" : "provider1@example.com".
  "schedule" : "New data provided on alternate rainy Fridays"
}
```

Example: Update "description" and remove "schedule"

To update "description" and remove "schedule", the following JSON can be used:

```
{
  "description" : "The Share Provider You Can Trust!(tm)",
  "schedule" : null
}
```

SET_RECIPIENT_LOG_LEVEL Procedure

Change the log level for an existing share recipient.

Syntax

```
PROCEDURE SET_RECIPIENT_LOG_LEVEL
(
  recipient_name       IN VARCHAR2,
  log_level            IN PLS_INTEGER,
  recipient_owner      IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
recipient_name	The local name of the share recipient.
log_level	Event logging level. For information on constants used for this parameter, see descriptions for **Log Level** in DBMS_SHARE Constants.
recipient_owner	The schema that owns the recipient.

SET_SHARE_LOG_LEVEL Procedure

Change the log level for an existing share.

Syntax

```
PROCEDURE SET_SHARE_LOG_LEVEL
(
  share_name           IN VARCHAR2,
  log_level            IN PLS_INTEGER,
  share_owner          IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
share_name	The name of the share.
log_level	Event logging level. For information on constants used for this parameter, see descriptions for **Log Level** in DBMS_SHARE Constants.
share_owner	The owner of the share.

SET_STORAGE_CREDENTIAL Procedure

Set the credential name used by the current user when it attempts to access the given storage.

Syntax

```
PROCEDURE SET_STORAGE_CREDENTIAL
(
    storage_link_name    IN VARCHAR2,
    credential_name      IN VARCHAR2,
    owner                IN VARCHAR2 := NULL,
    check_if_exists      IN BOOLEAN := TRUE,
    auto_commit          IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
storage_link_name	The name of a cloud storage link previously created using CREATE_CLOUD_STORAGE_LINK Procedure.

Parameter	Description
credential_name	The name of a local credential that gives access to the storage. The credentials used for share storage must be able to read, write, delete, and manage pre-authenticated requests. See Using Pre-Authenticated Requests
owner	The owner of the cloud storage link. Leave as NULL for the current user.
check_if_exists	If check_if_exists is TRUE, then the function will also confirm that the credential exists for the current user.
auto_commit	The default is TRUE. If TRUE, this transaction is committed. If FALSE, the user must commit the transaction. Changes are not visible until the commit takes place.

STOP_JOB Procedure

Attempt to stop a running share job. The procedure should return quickly, but it may take some time for the associated job to stop.

Syntax

```
PROCEDURE STOP_JOB
(
  share_job_id          IN NUMBER,
  share_job_owner       IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
share_job_id	The ID of the share job.
share_job_owner	The owner of the job. Defaults to the current schema.

UNPUBLISH_SHARE Procedure

Unpublish a share.

Syntax

```
PROCEDURE UNPUBLISH_SHARE
(
    share_name            IN VARCHAR2,
    out_share_job_id      IN OUT NOCOPY NUMBER,
    share_owner           IN VARCHAR2 := NULL,
    drop_all_versions     IN BOOLEAN := FALSE,
    restart_versions      IN BOOLEAN := FALSE
);
```

Parameters

Parameter	Description
share_name	The name of the share.

Parameter	Description
out_share_job_id	The ID of any share job that needs to be run to process this command.
share_owner	The owner of the share. Defaults to the current schema.
drop_all_versions	If set to TRUE, all existing versions along with all associated storage is dropped. If drop_all_versions is FALSE, then all existing versions continue to exist, but the share is not visible to recipients. A subsequent call to PUBLISH_SHARE or SET_CURRENT_SHARE_VERSION makes them visible again.
restart_versions	Restart version numbering. If FALSE, then the next published version number will be the same as it would have been if no versions had been dropped. If TRUE, then the next published version will be set to one more than the highest existing version. When used in conjunction with drop_all_versions, this has the effect of resetting the share to its original state. Note that this may confuse any existing delta clients, so it should be used with care.

UPDATE_DEFAULT_RECIPIENT_PROPERTY Procedure

Update the default recipient property values. This procedure requires the user to have admin privileges.

Syntax

```
PROCEDURE UPDATE_DEFAULT_RECIPIENT_PROPERTY
(
  recipient_property   IN VARCHAR2,
  new_value_vc         IN VARCHAR2
);
```

Parameters

Parameter	Description
recipient_property	The property to update. For information on constants used for this parameter, see descriptions for **Share Recipient Properties** in DBMS_SHARE Constants.
new_value_vc	The new property value.

UPDATE_DEFAULT_SHARE_PROPERTY Procedure

Update the default share property values.

Syntax

```
PROCEDURE UPDATE_DEFAULT_SHARE_PROPERTY
(
    share_property       IN VARCHAR2,
    new_value            IN VARCHAR2
);
```

Parameters

Parameter	Description
share_property	The property to update. For information on constants used for this parameter, see descriptions for **Share Properties** in DBMS_SHARE Constants. These properties can be read using the ALL_SHARE_DEFAULT_SETTINGS View.
new_value	The new property value.

UPDATE_RECIPIENT_PROPERTY Procedure

Update a property of an existing recipient.

Syntax

```
PROCEDURE UPDATE_RECIPIENT_PROPERTY
(
    recipient_name       IN VARCHAR2,
    recipient_property   IN VARCHAR2,
    new_value            IN VARCHAR2,
    recipient_owner      IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
recipient_name	The name of the recipient.
recipient_property	The property to update. These properties include: • PROP_RECIPIENT_EMAIL • PROP_RECIPIENT_DESCRIPTION • PROP_RECIPIENT_LOG_LEVEL • PROP_RECIPIENT_SHARING_ID • PROP_RECIPIENT_PROFILE_URL • PROP_RECIPIENT_PAR_TYPE • PROP_RECIPIENT_MIN_PAR_LIFETIME • PROP_RECIPIENT_PAR_LIFETIME • PROP_RECIPIENT_TOKEN_LIFETIME For information on constants used for this parameter, see descriptions for **Share Recipient Recipient Properties** in DBMS_SHARE Constants.
new_value	The new property value.
recipient_owner	The owner of the recipient. Defaults to the current schema.

UPDATE_SHARE_JOB_PROPERTY Procedure

Modify properties of a running share job. The procedure should return quickly, but it may take some time for the changes to take effect.

Syntax

```
PROCEDURE UPDATE_SHARE_JOB_PROPERTY
(
    share_job_id          IN NUMBER,
    share_property        IN VARCHAR2,
    new_value             IN VARCHAR2,
    share_job_owner       IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
share_job_id	The ID of the share job.
share_property	The property to update. For information on constants used for this parameter, see descriptions for **Share Job Properties** in DBMS_SHARE Constants.
new_value	The new property value.
share_job_owner	The owner of the job. Defaults to the current schema.

UPDATE_SHARE_PROPERTY Procedure

Update a property of an existing share.

Syntax

```
PROCEDURE UPDATE_SHARE_PROPERTY
(
    share_name            IN VARCHAR2,
    share_property        IN VARCHAR2,
    new_value             IN VARCHAR2,
    share_owner           IN VARCHAR2 := NULL,
    auto_commit           IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
share_name	The name of the share.
share_property	The property to update. For information on constants used for this parameter, see descriptions for **Share Properties** in DBMS_SHARE Constants.
new_value	The new property value.
share_owner	The owner of the share. Defaults to the current schema.

Parameter	Description
Auto_commit	If TRUE (the default), the changes are automatically committed. If FALSE, the user must commit the changes. Changes are visible externally after the commit takes place.

UPDATE_SHARE_TABLE_PROPERTY Procedure

Update the property value of an existing share table.

Syntax

```
PROCEDURE UPDATE_SHARE_TABLE_PROPERTY
(
    share_name           IN VARCHAR2,
    share_table_name     IN VARCHAR2,
    share_table_property IN VARCHAR2,
    new_value            IN VARCHAR2,
    share_schema_name    IN VARCHAR2 := NULL,
    share_owner          IN VARCHAR2 := NULL,
    auto_commit          IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
share_name	The name of the share.
share_table_name	The name of the share table.
share_table_property	The property to update. These properties include: • PROP_SHARE_TABLE_SPLIT_METHOD • PROP_SHARE_TABLE_SPLIT_COLUMNS • PROP_SHARE_TABLE_ORDER_COLUMNS • PROP_SHARE_TABLE_SHARE_COLUMNS • PROP_SHARE_TABLE_SPLIT_SIZE • PROP_SHARE_TABLE_GATHER_STATS • PROP_SHARE_TABLE_SPLIT_ROWS • PROP_SHARE_TABLE_FLASHBACK
new_value	The new property value.
share_schema_name	The name of the share schema (defaults to uppercase of current user).
share_owner	The owner of the share. Defaults to the current schema.
Auto_commit	If TRUE (the default), the changes are automatically committed. If FALSE, the user must commit the changes. Changes are visible externally after the commit takes place.

VALIDATE_CREDENTIAL Function

Validate a credential name, converting it to canonical form first if required.

The procedure raises an exception if the name is not a valid Oracle identifier. The `credential_name` is returned without double quotes, as it would appear in the `CREDENTIAL_NAME` column of the `USER_CREDENTIALS` view.

Syntax

```
FUNCTION VALIDATE_CREDENTIAL
(
        credential_name      IN VARCHAR2,
        check_if_exists      IN BOOLEAN := TRUE
)
RETURN VARCHAR2;
```

Parameters

Parameter	Description
credential_name	The name of the credential to validate in standard database form, with double quotes if the name is not a simple identifier.
check_if_exists	If TRUE, then the function also confirms that the credential exists for the current user.

VALIDATE_SHARE_STORAGE Procedure

Check to see if the given storage is suitable for versioned shares.

Syntax

The procedure syntax including the validation_results output parameter.

```
 PROCEDURE VALIDATE_SHARE_STORAGE
(
    storage_link_name    IN VARCHAR2,
    validation_results   IN OUT NOCOPY VARCHAR2,
    run_storage_tests    IN BOOLEAN := TRUE,
    storage_link_owner   IN VARCHAR2 := NULL
);
```

The procedure syntax not including the validation_results output parameter.

```
PROCEDURE VALIDATE_SHARE_STORAGE
(
    storage_link_name    IN VARCHAR2,
    run_storage_tests    IN BOOLEAN := TRUE,
    storage_link_owner   IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
storage_link_name	The cloud storage link name.
validation_results	(Optional input and output) Validation result details returned in JSON form. The validation_results include the results for each separate test.
run_storage_tests	Run tests to validate the storage. If TRUE (the default), the procedure tests READ, WRITE, DELETE, and PREAUTHENTICATED REQUESTS.
storage_link_owner	The cloud storage link owner.

Example: Validation results

```
{
  "READ":"PASSSED",
  "WRITE":"PASSSED",
  "CREATE_PAR":"PASSSED",
  "DELETE_PAR":"PASSSED",
  "DELETE":"PASSSED"
}
```

WAIT_FOR_JOB Procedure

This procedure waits until the specified share job is complete.

Syntax

```
PROCEDURE WAIT_FOR_JOB
(
    share_job_id         IN NUMBER,
    completed            IN OUT NOCOPY BOOLEAN,
    maximum_wait         IN NUMBER := NULL
);
```

Parameters

Parameter	Description
share_job_id	The ID of the share job.
completed	Job completion indicator.
maximum_wait	The maximum wait period, in seconds. A NULL value implies no limit.

Summary of Share Consumer Subprograms

This table lists the DBMS_SHARE package procedures and functions used to consume shares.

Subprograms	Description
ASSERT_SHARING_ID Procedure	Run basic validation checks against a sharing id and return one in canonical form.
CREATE_CREDENTIALS Procedure and Function	Create a credential containing the bearer token from a delta profile.
CREATE_OR_REPLACE_SHARE_LINK Procedure	Subscribe to share from a registered share provider.
CREATE_OR_REPLACE_ORACLE_SHARE_PROVIDER Procedure	Subscribe to an Oracle share provider, with a local name.
CREATE_ORACLE_SHARE_PROVIDER Procedure	Subscribe to an Oracle share provider, with a local name.
CREATE_SHARE_LINK Procedure	Subscribe to share from a registered share provider.
CREATE_SHARE_LINK_VIEW Procedure	Create or replace a named view that gives access to a remote shared table.

Subprograms	Description
CREATE_SHARE_PROVIDER Procedure	Subscribe to a delta share provider.
DISCOVER_AVAILABLE_SHARES Function	Return one `SHARE_AVAILABLE_SHARES_ROW` for each available table from the subscribed share providers.
DISCOVER_AVAILABLE_TABLES Function	Return one `SHARE_AVAILABLE_TABLES_ROW` for each available table from the subscribed share providers or from an explicit delta endpoint.
DROP_SHARE_LINK Procedure	Drop a share link created by the `CREATE_SHARE_LINK` procedure.
DROP_SHARE_PROVIDER Procedure	Drop a subscription to a share provider.
ENABLE_DELTA_ENDPOINT Procedure	Create necessary ACLs that allow the specified user to connect to a delta endpoint.
FLUSH_SHARE_LINK_CACHE Procedure	Flush the cache of shares for a given share link.
FLUSH_SHARE_PROVIDER_CACHE Procedure	Flush the cache of shares for a given share provider.
GENERATE_SHARE_LINK_SELECT Procedure and Function	Generate a SELECT statement that returns data from a shared table.
GET_ORACLE_SHARE_LINK_INFO Function	Retrieve the cloud link name and namespace for an Oracle-to-Oracle share.
GET_SHARE_LINK_INFO Procedure	Get the endpoint(s), share type and share name along with any additional JSON metadata for a share link.
GET_SHARE_PROVIDER_CREDENTIAL Procedure	Get the credential name to be used by the current user when it attempts to access the given delta share provider.
GET_SHARE_PROVIDER_INFO Procedure	Get the endpoint string(s) and share type along with any additional JSON metadata for a share provider.
GET_SHARING_ID Function	Return an identifier that can be used as the `sharing_id` in the `CREATE_SHARE_RECIPIENT` procedure.
OPEN_SHARE_LINK_CURSOR Procedure	Open a cursor that returns data from a shared table.
REFRESH_BEARER_TOKEN_CREDENTIAL Procedure	Update one or more credentials created by `CREATE_BEARER_TOKEN_CREDENTIAL` or `CREATE_CREDENTIALS`.
RENAME_CLOUD_STORAGE_LINK Procedure	Rename a registered cloud storage link.
RENAME_SHARE_LINK Procedure	Rename a registered share link.
RENAME_SHARE_PROVIDER Procedure	Rename a registered share provider.
REMOVE_SHARE_SCHEMA Procedure	Remove a schema and all its contents from a share.
SET_SHARE_LINK_METADATA Procedure	Set the additional JSON metadata for the share link.
SET_SHARE_PROVIDER_CREDENTIAL Procedure	Set the credential name to access the given share provider.
SET_SHARE_PROVIDER_METADATA Procedure	Set additional JSON metadata for the share provider.
UPDATE_BEARER_TOKEN_CREDENTIAL Procedure	Modify an attribute of a credential created by `CREATE_CREDENTIALS` or `CREATE_BEARER_TOKEN_CREDENTIAL`.

- ASSERT_SHARING_ID Procedure
  Run basic validation checks against a sharing id and return one in canonical form. An exception is raised if the id is obviously invalid.
- CREATE_CREDENTIALS Procedure and Function
  Create a credential containing the bearer token from a delta sharing profile. The standard type, version 1, specifies an endpoint and a single long term bearer token.
- CREATE_OR_REPLACE_SHARE_LINK Procedure
  Subscribe to share from a registered share provider.
- CREATE_OR_REPLACE_ORACLE_SHARE_PROVIDER Procedure
  Subscribe to an Oracle share provider, with a local name.
- CREATE_ORACLE_SHARE_PROVIDER Procedure
  Subscribe to an Oracle share provider, with a local name.
- CREATE_SHARE_LINK Procedure
  Subscribe to share from a registered share provider. The available share names can be found by calling `DISCOVER_AVAILABLE_SHARES`.
- CREATE_SHARE_LINK_VIEW Procedure
  Create or replace a named view that gives access to a remote shared table.
- CREATE_SHARE_PROVIDER Procedure
  Subscribe to a delta share provider.
- DISCOVER_AVAILABLE_SHARES Function
  Return one `SHARE_AVAILABLE_SHARES_ROW` for each available table from the subscribed share providers.
- DISCOVER_AVAILABLE_TABLES Function
  Return one `SHARE_AVAILABLE_TABLES_ROW` for each available table from the subscribed share providers.
- DROP_SHARE_LINK Procedure
  Drop a share link created by the `CREATE_SHARE_LINK` procedure.
- DROP_SHARE_PROVIDER Procedure
  Drop a subscription to a share provider.
- ENABLE_DELTA_ENDPOINT Procedure
  Create necessary ACLs that allow the specified user to connect to a delta endpoint. Admin privileges are required for this procedure.
- FLUSH_SHARE_LINK_CACHE Procedure
  Flush the cache of shares for a given share link. The list of shares for the remote endpoints is fetched instead of relying on cached values.
- FLUSH_SHARE_PROVIDER_CACHE Procedure
  Flush the cache of shares for a given share provider. The list of shares for the remote endpoints is fetched instead of relying on cached values.
- GENERATE_SHARE_LINK_SELECT Procedure and Function
  Generate a SELECT statement that returns data from a shared table.
- GET_ORACLE_SHARE_LINK_INFO Function
  Retrieve the cloud link name and namespace for an Oracle-to-Oracle share.
- GET_SHARE_LINK_INFO Procedure
  Get the endpoint(s), share type and share name along with any additional JSON metadata for a share link.

- GET_SHARE_PROVIDER_CREDENTIAL Procedure
  Get the credential name to be used by the current user when it attempts to access the given delta share provider.
- GET_SHARE_PROVIDER_INFO Procedure
  Get the endpoint string(s) and share type along with any additional JSON metadata for a share provider. For ORACLE share providers, the Oracle provider ID is returned in the endpoint argument.
- GET_SHARING_ID Function
  Return an identifier that can be used as the `sharing_id` in the `CREATE_SHARE_RECIPIENT` procedure. This function can be used to share data between two users, the "provider" and the "recipient", in different databases.
- OPEN_SHARE_LINK_CURSOR Procedure
  Open a cursor that returns data from a shared table.
- REFRESH_BEARER_TOKEN_CREDENTIAL Procedure
  Update one or more credentials created by `CREATE_BEARER_TOKEN_CREDENTIAL` or `CREATE_CREDENTIALS` by calling the registered token endpoints and fetching new bearer tokens. Note this procedure is called automatically by a scheduler job, `ADP$BEARER_REFRESH_JOB`, that runs every 50 minutes.
- RENAME_CLOUD_STORAGE_LINK Procedure
  Rename a registered cloud storage link.
- RENAME_SHARE_PROVIDER Procedure
  Rename a registered share provider.
- REMOVE_SHARE_SCHEMA Procedure
  Remove a schema and all its contents from a share.
- SET_SHARE_LINK_METADATA Procedure
  Set the additional JSON metadata for the share link.
- SET_SHARE_PROVIDER_CREDENTIAL Procedure
  Set the credential name to be used by the current user when it attempts to access the given share provider.
- SET_SHARE_PROVIDER_METADATA Procedure
  Set additional JSON metadata for the share provider.
- UPDATE_BEARER_TOKEN_CREDENTIAL Procedure
  Modify an attribute of a credential created by `CREATE_CREDENTIALS` or `CREATE_BEARER_TOKEN_CREDENTIAL`.

ASSERT_SHARING_ID Procedure

Run basic validation checks against a sharing id and return one in canonical form. An exception is raised if the id is obviously invalid.

Syntax

```
PROCEDURE ASSERT_SHARING_ID
(
    sharing_id              IN OUT NOCOPY VARCHAR2,
    out_sharing_id_type     IN OUT NOCOPY VARCHAR2
);
```

Parameters

Parameter	Description
sharing_id	The id to check.
out_sharing_id_type	The type of the id, if valid. For example, TENANCY or DATABASE.

CREATE_CREDENTIALS Procedure and Function

Create a credential containing the bearer token from a delta sharing profile. The standard type, version 1, specifies an endpoint and a single long term bearer token.

Syntax

```
PROCEDURE CREATE_CREDENTIALS
(
    credential_base_name IN VARCHAR2,
    delta_profile        IN CLOB,
    out_credential_name  IN OUT NOCOPY VARCHAR2
);
```

Below is the functional version of the create_credentials that returns the name of the new credentials in JSON form.

```
FUNCTION CREATE_CREDENTIALS
(
   credential_base_name IN VARCHAR2,
   delta_profile        IN CLOB
)
RETURN CLOB;
```

Parameters

Parameter	Description
credential_base_name	The base name of the credential(s) to create.

Parameter	Description
delta_profile	The delta sharing profile, in JSON format, obtained from the share provider. { "shareCredentialsVersion": 1, "endpoint": "https://<endpoint>", "bearerToken": "<token>", "expirationTime": "...", } The profile may also, optionally, include a tokenEndpoint property along with a clientID and clientSecret. { "shareCredentialsVersion": 1, "endpoint": "https://<endpoint>", "bearerToken": "<token>", "expirationTime": "...", "tokenEndpoint": "https://<token endpoint>", "clientID": "<client id>", "clientSecret": "<client secret>" } See Profile File Format and Bearer Token for further information.
out_credential_name	The name of the newly created bearer token credential.

CREATE_OR_REPLACE_SHARE_LINK Procedure

Subscribe to share from a registered share provider.

Syntax

```
PROCEDURE CREATE_OR_REPLACE_SHARE_LINK
(
    share_link_name        IN VARCHAR2,
    share_provider         IN VARCHAR2,
    share_name             IN VARCHAR2,
    provider_owner         IN VARCHAR2 := NULL,
    link_owner             IN VARCHAR2 := NULL,
    use_default_credential IN BOOLEAN := TRUE,
    metadata               IN SYS.JSON_OBJECT_T := NULL,
    auto_commit            IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
share_link_name	The name of the share link that will be created. The available share names can be found by calling DISCOVER_AVAILABLE_SHARES Function.
share_provider	The name of the registered share provider.

Parameter	Description
share_name	The name of the share from the share provider.
provider_owner	The owner of the share provider. Defaults to the current schema.
link_owner	The owner of the share link. Defaults to the current schema.
use_default_credential	If TRUE, use the same credentials as the share provider.
metadata	Optional metadata to associate with the share link.
auto_commit	If TRUE (the default), this procedure call commits changes that are not visible externally until the commit takes place. If FALSE, the user must COMMIT after running this call in order to make the change visible.

CREATE_OR_REPLACE_ORACLE_SHARE_PROVIDER Procedure

Subscribe to an Oracle share provider, with a local name.

It will then appear in ALL_SHARE_PROVIDERS View with `RECIPIENT_TYPE = 'ORACLE'`.

> **Note:**
>
> Use the `SET_STORAGE_CREDENTIAL` procedure to add a credential to the storage link. See SET_STORAGE_CREDENTIAL Procedure.

Syntax

```
PROCEDURE CREATE_OR_REPLACE_CLOUD_STORAGE_LINK
(
    storage_link_name    IN VARCHAR2,
    uri                  IN VARCHAR2,
    owner                IN VARCHAR2 := NULL,
    metadata             IN SYS.JSON_OBJECT_T := NULL,
    auto_commit          IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
storage_link_name	The name of the cloud storage link. The name of the link should follow standard Oracle naming conventions. See Database Object Names and Qualifiers
uri	The URI of the storage bucket. The URI should be of the form `https://objectstorage.region.oraclecloud.com/n/namespace-string/b/bucket/o`
owner	The owner of the storage link. Defaults to the current schema.
metadata	An optional JSON metadata document containing additional information.
auto_commit	If TRUE, the changes automatically commit after creating the link. The default is TRUE.

CREATE_ORACLE_SHARE_PROVIDER Procedure

Subscribe to an Oracle share provider, with a local name.

It will then appear in ALL_SHARE_PROVIDERS View with `RECIPIENT_TYPE = 'ORACLE'`.

Syntax

```
PROCEDURE CREATE_ORACLE_SHARE_PROVIDER
(
    oracle_provider_id   IN VARCHAR2,
    provider_name        IN VARCHAR2,
    owner                IN VARCHAR2 := NULL,
    metadata             IN SYS.JSON_OBJECT_T := NULL,
    auto_commit          IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
`oracle_provider_id`	The provider ID obtained from the `ALL_AVAILABLE_ORACLE_SHARE_PROVIDERS` view. See ALL_AVAILABLE_ORACLE_SHARE_PROVIDERS View.
`provider_name`	A local name for the provider
`owner`	The owner of the new share provider. Leave as NULL for the current user.
`metadata`	Optional JSON metadata to associate with the provider.
`auto_commit`	If TRUE, the changes automatically commit after creating the link. The default is TRUE.

CREATE_SHARE_LINK Procedure

Subscribe to share from a registered share provider. The available share names can be found by calling `DISCOVER_AVAILABLE_SHARES`.

Syntax

```
PROCEDURE CREATE_SHARE_LINK
(
    share_link_name      IN VARCHAR2,
    share_provider       IN VARCHAR2,
    share_name           IN VARCHAR2,
    provider_owner       IN VARCHAR2 := NULL,
    link_owner           IN VARCHAR2 := NULL,
    use_default_credential IN BOOLEAN := TRUE,
    metadata             IN SYS.JSON_OBJECT_T := NULL,
    auto_commit          IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
share_link_name	The name of the share link to create. This should follow standard Oracle naming conventions and does not need to be the same as the share_name parameter.
share_provider	The name of the share provider, listed in ALL_SHARE_PROVIDERS, that shared the data.
share_name	The name of the share, as defined by the share provider.
provider_owner	The schema that subscribed the share provider, as listed in ALL_SHARE_PROVIDER. Leave as null to default to the current user, which is typical.
link_owner	The owner of the new share link. Leave as null to default to the current user, which is typical.
use_default_credential	Set to TRUE to use the same credential for both the share link and the share provider.
metadata	Optional JSON metadata to associated with the share link.
auto_commit	If TRUE, the changes automatically commit after creating the link. The default is TRUE.

CREATE_SHARE_LINK_VIEW Procedure

Create or replace a named view that gives access to a remote shared table.

Syntax

```
PROCEDURE CREATE_SHARE_LINK_VIEW
(
    view_name            IN VARCHAR2,
    share_link_name      IN VARCHAR2,
    share_schema_name    IN VARCHAR2,
    share_table_name     IN VARCHAR2,
    view_owner           IN VARCHAR2 := NULL,
    share_link_owner     IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
view_name	The name of the new view.
share_link_name	The name of the share link.
share_schema_name	The name of the shared schema.
share_table_name	The name of the shared table.
view_owner	The view owner. Defaults to the current schema.
share_link_owner	The link owner. Defaults to the current schema.

CREATE_SHARE_PROVIDER Procedure

Subscribe to a delta share provider.

Syntax

```
PROCEDURE CREATE_SHARE_PROVIDER
(
  provider_name         IN VARCHAR2,
  endpoint              IN VARCHAR2,
  token_endpoint        IN VARCHAR2 := NULL,
  share_type            IN VARCHAR2 := 'DELTA',
  owner                 IN VARCHAR2 := NULL,
  metadata              IN SYS.JSON_OBJECT_T := NULL,
  auto_commit           IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
provider_name	The local name of this share provider.
endpoint	The delta endpoint, from the delta sharing profile.
token_endpoint	This parameter is ignored.
share_type	The type of share provider. Leave this as DELTA.
owner	The owner of the share provider. Defaults to the current schema.
metadata	Optional JSON metadata to associate with the share provider.
auto_commit	If TRUE (the default), this procedure call commits changes that are not visible externally until the commit takes place. If FALSE, the user must COMMIT after running this call in order to make the change visible.

DISCOVER_AVAILABLE_SHARES Function

Return one SHARE_AVAILABLE_SHARES_ROW for each available table from the subscribed share providers.

Syntax

```
FUNCTION DISCOVER_AVAILABLE_SHARES
(
    share_provider       IN VARCHAR2,
    owner                IN VARCHAR2 := NULL
) RETURN share_available_shares_tbl PIPELINED;
```

Parameters

Parameter	Description
share_provider	The name of the share provider.
owner	The owner of the share provider. Defaults to the current schema.

Example: Discover a list of shares available from a given provider

```
SQL> select available_share_name
  2    from dbms_share.discover_available_shares('share_prov')
  3   order by available_share_name;
```

```
AVAILABLE_SHARE_NAME
--------------------------------------------------------------------------------
--
BURLINGTON_EXPEDITION_2022

EGYPT_EXPEDITION_2022
```

DISCOVER_AVAILABLE_TABLES Function

Return one `SHARE_AVAILABLE_TABLES_ROW` for each available table from the subscribed share providers.

Syntax

```
FUNCTION DISCOVER_AVAILABLE_TABLES
(
  share_provider       IN VARCHAR2 := NULL,
  share_name           IN VARCHAR2 := NULL,
  owner                IN VARCHAR2 := NULL,
  endpoint             IN VARCHAR2 := NULL,
  credential_name      IN VARCHAR2 := NULL
) RETURN share_available_tables_tbl PIPELINED;
```

Parameters

Parameter	Description
`share_provider`	An optional share provider name. If NULL, search all subscribed share providers.
`share_name`	An optional share name. If NULL, search all discovered shares.
`owner`	The owner of the share provider. Defaults to the current schema.
`endpoint`	An optional delta endpoint.
`credential_name`	An optional bearer token credential to access the endpoint.

Example: List shares available from all subscribed share providers

```
SQL> select * from dbms_share.discover_available_tables()
  2  order by share_name, schema_name, table_name;

 PROVIDER_NAME              PROVIDER_OWNER  SHARE_NAME
------------------------- -------------- ------------------------------
SCHEMA_NAME                TABLE_NAME
------------------------- -------------------------
My Test Oracle Provider    ADP_SHARE_TEST  BURLINGTON_EXPEDITION_2022
SH                         COUNTRIES

My Test Oracle Provider    ADP_SHARE_TEST  BURLINGTON_EXPEDITION_2022
SH                         SH_COUNTRIES

My Test Oracle Provider    ADP_SHARE_TEST  EGYPT_EXPEDITION_2022
SHARED_SCHEMA              SHARED_VIEW_1
```

```
My Test Oracle Provider   ADP_SHARE_TEST  EGYPT_EXPEDITION_2022
SHARED_SCHEMA             SHARED_VIEW_2
```

Example: List tables available from an unsubscribed endpoint

```
SQL> exec dbms_cloud.create_credential('MY_CRED', 'BEARER_TOKEN', '123456')
PL/SQL procedure successfully completed.
SQL> column share_name format a13
SQL> column table_name format a20
SQL> column schema_name format a10
SQL> select share_name, schema_name, table_name
2   from dbms_share.discover_available_tables(
3     endpoint=>'https://my_endpoint',
4     credential_name=>'MY_CRED')
5   order by 1, 2, 3;

SHARE_NAME SCHEMA_NAM TABLE_NAME
------------ ---------- --------------------
DELTA_SHARING DEFAULT BOSTON-HOUSING
DELTA_SHARING DEFAULT COVID_19_NYT
DELTA_SHARING DEFAULT FLIGHT-ASA_2008
DELTA_SHARING DEFAULT LENDING_CLUB
DELTA_SHARING DEFAULT NYCTAXI_2019
DELTA_SHARING DEFAULT NYCTAXI_2019_PART
DELTA_SHARING DEFAULT OWID-COVID-DATA

7 rows selected.
```

DROP_SHARE_LINK Procedure

Drop a share link created by the `CREATE_SHARE_LINK` procedure.

See CREATE_SHARE_LINK Procedure for further information.

Syntax

```
PROCEDURE DROP_SHARE_LINK
(
    link_name            IN VARCHAR2,
    link_owner           IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
`link_name`	The name of the share link to drop.
`link_owner`	The owner of the share link to drop. Leave as null for the current schema.

DROP_SHARE_PROVIDER Procedure

Drop a subscription to a share provider.

Syntax

```
PROCEDURE DROP_SHARE_PROVIDER
(
    provider_name        IN VARCHAR2,
    owner                IN VARCHAR2 := NULL,
    drop_credentials     IN BOOLEAN := FALSE
);
```

Parameters

Parameter	Description
provider_name	The name of the share provider to drop.
owner	The owner of the share provider to drop. Defaults to the current schema.
drop_credentials	If TRUE, any credentials associated with the provider are dropped. If FALSE, credentials are not dropped.

ENABLE_DELTA_ENDPOINT Procedure

Create necessary ACLs that allow the specified user to connect to a delta endpoint. Admin privileges are required for this procedure.

Syntax

```
PROCEDURE ENABLE_DELTA_ENDPOINT
(
    schema_name          IN VARCHAR2,
    delta_profile        IN CLOB,
    enabled              IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
schema_name	The schema to enable or disable.
delta_profile	The delta profile. Only the endpoint and tokenEndpoint are required.
enabled	TRUE to enable and FALSE to disable.

FLUSH_SHARE_LINK_CACHE Procedure

Flush the cache of shares for a given share link. The list of shares for the remote endpoints is fetched instead of relying on cached values.

Syntax

```
PROCEDURE FLUSH_SHARE_LINK_CACHE
(
```

```
    link_name            IN VARCHAR2,
    owner                IN VARCHAR2 := NULL,
    auto_commit          IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
link_name	The name of the share link.
owner	The owner of the share link. Defaults to the current schema.
auto_commit	If TRUE, the changes are automatically committed. The default is TRUE.

FLUSH_SHARE_PROVIDER_CACHE Procedure

Flush the cache of shares for a given share provider. The list of shares for the remote endpoints is fetched instead of relying on cached values.

Syntax

```
PROCEDURE FLUSH_SHARE_PROVIDER_CACHE
(
  provider_name        IN VARCHAR2,
  owner                IN VARCHAR2 := NULL,
  auto_commit          IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
provider_name	The name of the share provider.
owner	The owner of the share provider. Defaults to the current schema.
auto_commit	If TRUE, the changes are automatically committed. The default is TRUE.

GENERATE_SHARE_LINK_SELECT Procedure and Function

Generate a SELECT statement that returns data from a shared table.

Syntax

Procedure version of `GENERATE_SHARE_LINK_SELECT`.

```
PROCEDURE GENERATE_SHARE_LINK_SELECT
(
    share_link_name      IN VARCHAR2,
    share_schema_name    IN VARCHAR2,
    share_table_name     IN VARCHAR2,
    stmt                 IN OUT NOCOPY CLOB,
    share_link_owner     IN VARCHAR2 := NULL
);
```

Function version of `GENERATE_SHARE_LINK_SELECT`.

```
 FUNCTION GENERATE_SHARE_LINK_SELECT
(
    share_link_name      IN VARCHAR2,
    share_schema_name    IN VARCHAR2,
    share_table_name     IN VARCHAR2,
    share_link_owner     IN VARCHAR2 := NULL
)
RETURN CLOB;
```

Parameters

Parameter	Description
`share_link_name`	The name of the share link.
`share_schema_name`	The name of the shared schema.
`share_table_name`	The name of the shared table.
`stmt`	The generated select statement.
`share_link_owner`	The link owner. Defaults to the current schema.

GET_ORACLE_SHARE_LINK_INFO Function

Retrieve the cloud link name and namespace for an Oracle-to-Oracle share.

Syntax

```
FUNCTION GET_ORACLE_SHARE_LINK_INFO
(
  oracle_provider_id   IN VARCHAR2,
  share_name           IN VARCHAR2,
  share_schema_name    IN VARCHAR2,
  share_table_name     IN VARCHAR2
)
RETURN CLOB;
```

Parameters

Parameter	Description
`oracle_provider_id`	The Oracle Provider ID from `ALL_AVAILABLE_ORACLE_SHARE_PROVIDERS`.
`share_name`	The name of a share from `ALL_AVAILABLE_ORACLE_SHARE_PROVIDERS.SHARES`.
`share_schema_name`	The name of a share schema from `ALL_AVAILABLE_ORACLE_SHARE_PROVIDERS.SHARES`.
`share_table_name`	The name of a table from `ALL_AVAILABLE_ORACLE_SHARE_PROVIDERS.SHARES`.

Return

The return value is a JSON object containing three properties: schema, table, and dblink. The caller can use these three properties to fetch data using a query of the following form:

```
SELECT *
FROM <schema>.<table>@<dblink>
```

GET_SHARE_LINK_INFO Procedure

Get the endpoint(s), share type and share name along with any additional JSON metadata for a share link.

Syntax

```
PROCEDURE GET_SHARE_LINK_INFO
(
    link_name            IN VARCHAR2,
    endpoint             IN OUT NOCOPY VARCHAR2,
    share_type           IN OUT NOCOPY VARCHAR2,
    share_name           IN OUT NOCOPY VARCHAR2,
    token_endpoint       IN OUT NOCOPY VARCHAR2,
    metadata             IN OUT NOCOPY BLOB,
    link_owner           IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
link_name	The name of a share link (previously created by CREATE_SHARE_LINK).
endpoint	OUT: The endpoint, if any, associated with the share link.
share_type	OUT: The type of share link (for example, DELTA).
share_name	OUT: The name of the linked share.
token_endpoint	This parameter is no longer used.
metadata	OUT: Optional metadata associated with the share link.
link_owner	The owner of the share link. Defaults to the current schema.

GET_SHARE_PROVIDER_CREDENTIAL Procedure

Get the credential name to be used by the current user when it attempts to access the given delta share provider.

Syntax

```
PROCEDURE GET_SHARE_PROVIDER_CREDENTIAL
(
    provider_name        IN VARCHAR2,
    share_credential     IN OUT NOCOPY VARCHAR2,
    get_token_credential IN OUT NOCOPY VARCHAR2,
    owner                IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
provider_name	The name of the share provider.
share_credential	OUT: The name of the credential associated with the provider. The credential name is returned without double quotes, as it would appear in the CREDENTIAL_NAME column of the USER_CREDENTIALS view. See ALL_CREDENTIALS View.
get_token_credential	This parameter is not used.
owner	The owner is the name of the schema where the share provider was registered, not the owner of the credential. Defaults to the current schema.

GET_SHARE_PROVIDER_INFO Procedure

Get the endpoint string(s) and share type along with any additional JSON metadata for a share provider. For ORACLE share providers, the Oracle provider ID is returned in the endpoint argument.

Syntax

```
PROCEDURE GET_SHARE_PROVIDER_INFO
(
    provider_name        IN VARCHAR2,
    endpoint             IN OUT NOCOPY VARCHAR2,
    share_type           IN OUT NOCOPY VARCHAR2,
    token_endpoint       IN OUT NOCOPY VARCHAR2,
    metadata             IN OUT NOCOPY BLOB,
    owner                IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
provider_name	The name of the share provider.
endpoint	The delta endpoint.
share_type	The share type: DELTA or ORACLE.
token_endpoint	This parameter is not used.
metadata	The optional metadata that was associated with the share provider.
owner	The owner of the share provider. Defaults to the current schema.

GET_SHARING_ID Function

Return an identifier that can be used as the sharing_id in the CREATE_SHARE_RECIPIENT procedure. This function can be used to share data between two users, the "provider" and the "recipient", in different databases.

See CREATE_SHARE_RECIPIENT Procedure for further information.

Syntax

```
FUNCTION GET_SHARING_ID
(
    sharing_id_type      IN VARCHAR2 := SHARING_ID_TYPE_DATABASE
)
  RETURN VARCHAR2;
```

Parameters

Parameter	Description
sharing_id_type	The type of sharing ID.

Usage

The flow is as follows:

1. The recipient calls `DBMS_SHARE.GET_SHARING_ID` to get a unique identifier.
2. The recipient sends this identifier (e.g. via email) to the provider.
3. The provider calls `DBMS_SHARE.CREATE_SHARE_RECIPIENT`, passing in the identifier as `sharing_id`.
4. The provider calls `DBMS_SHARE.GRANT_TO_RECIPIENT` to give the recipient access to shared data.

The `sharing_id_type` parameter is used to specify which database users can access the share following the above sequence.

- DATABASE The share will be visible to any admin user in the database where `GET_SHARING_ID` was called.
- COMPARTMENT The share will be visible to any admin user in any database in the same compartment where `GET_SHARING_ID` was called.
- TENANCY The share will be visible to any admin user in any database in the same tenancy where `GET_SHARING_ID` was called.
- REGION The share will be visible to any admin user in any database in the same region where `GET_SHARING_ID` was called.

OPEN_SHARE_LINK_CURSOR Procedure

Open a cursor that returns data from a shared table.

Syntax

```
PROCEDURE OPEN_SHARE_LINK_CURSOR
(
    share_link_name      IN VARCHAR2,
    share_schema_name    IN VARCHAR2,
    share_table_name     IN VARCHAR2,
    table_cursor         IN OUT NOCOPY SYS_REFCURSOR,
    share_link_owner     IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
share_link_name	The name of the share link.
share_schema_name	The name of the shared schema.
share_table_name	The name of the shared table.
table_cursor	The cursor.
share_link_owner	The link owner. Defaults to current schema.

REFRESH_BEARER_TOKEN_CREDENTIAL Procedure

Update one or more credentials created by CREATE_BEARER_TOKEN_CREDENTIAL or CREATE_CREDENTIALS by calling the registered token endpoints and fetching new bearer tokens. Note this procedure is called automatically by a scheduler job, ADP$BEARER_REFRESH_JOB, that runs every 50 minutes.

See CREATE_BEARER_TOKEN_CREDENTIAL Procedure or CREATE_CREDENTIALS Procedure and Function for further information.

Syntax

```
PROCEDURE REFRESH_BEARER_TOKEN_CREDENTIAL
(
    credential_name           IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
credential_name	The name of the credential to refresh.

RENAME_CLOUD_STORAGE_LINK Procedure

Rename a registered cloud storage link.

Syntax

```
PROCEDURE RENAME_CLOUD_STORAGE_LINK
(
    old_name              IN VARCHAR2,
    new_name              IN VARCHAR2,
    owner                 IN VARCHAR2 := NULL,
    auto_commit           IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
old_name	The current name of the cloud storage link.

Parameter	Description
new_name	The new name of the cloud storage link.
owner	The owner of the cloud storage link. Defaults to the current schema.
auto_commit	If TRUE, the changes are automatically committed. The default is TRUE.

RENAME_SHARE_PROVIDER Procedure

Rename a registered share provider.

Syntax

```
PROCEDURE RENAME_SHARE_PROVIDER
(
  old_name              IN VARCHAR2,
  new_name              IN VARCHAR2,
  owner                 IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
old_name	The current name of the share provider.
new_name	The new name of the share provider.
owner	The owner of the share provider. Defaults to the current schema.

REMOVE_SHARE_SCHEMA Procedure

Remove a schema and all its contents from a share.

Syntax

```
PROCEDURE REMOVE_SHARE_SCHEMA
(
    share_name            IN VARCHAR2,
    schema_name           IN VARCHAR2,
    share_owner           IN VARCHAR2 := NULL,
    auto_commit           IN BOOLEAN := FALSE
);
```

Parameters

Parameter	Description
share_name	The name of the share.
schema_name	The name of the schema to remove.
share_owner	The owner of the share.
auto_commit	If TRUE, the changes are automatically committed. The default is FALSE.

SET_SHARE_LINK_METADATA Procedure

Set the additional JSON metadata for the share link.

Syntax

```
PROCEDURE SET_SHARE_LINK_METADATA
(
    link_name            IN VARCHAR2,
    metadata             IN SYS.JSON_OBJECT_T,
    replace_existing     IN BOOLEAN := FALSE,
    link_owner           IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
link_name	The name of the share link.
metadata	The new metadata.
replace_existing	If TRUE, then all existing metadata is replaced by the new version. If FALSE, then the new value is merged with any existing metadata.
link_owner	The owner of the share link. Defaults to the current schema.

SET_SHARE_PROVIDER_CREDENTIAL Procedure

Set the credential name to be used by the current user when it attempts to access the given share provider.

Syntax

```
PROCEDURE SET_SHARE_PROVIDER_CREDENTIAL
(
    provider_name        IN VARCHAR2,
    share_credential     IN VARCHAR2,
    get_token_credential IN VARCHAR2 := NULL,
    owner                IN VARCHAR2 := NULL,
    check_if_exists      IN BOOLEAN := TRUE
);
```

Parameters

Parameter	Description
provider_name	The name of the share provider.
share_credential	The bearer token credential.
get_token_credential	This argument is ignored.
owner	The owner of the share provider. Defaults to the current schema.
check_if_exists	If TRUE (default), then validate that the credential exists.

SET_SHARE_PROVIDER_METADATA Procedure

Set additional JSON metadata for the share provider.

Syntax

```
PROCEDURE SET_SHARE_PROVIDER_METADATA
(
    provider_name        IN VARCHAR2,
    metadata             IN SYS.JSON_OBJECT_T,
    replace_existing     IN BOOLEAN := FALSE,
    owner                IN VARCHAR2 := NULL
);
```

Parameters

Parameter	Description
provider_name	The name of the share provider.
metadata	The new metadata.
replace_existing	If TRUE, then all existing metadata is replaced by the new version. If FALSE, the new value is merged with any existing metadata.
owner	The owner of the share provider. Defaults to the current schema.

UPDATE_BEARER_TOKEN_CREDENTIAL Procedure

Modify an attribute of a credential created by CREATE_CREDENTIALS or CREATE_BEARER_TOKEN_CREDENTIAL.

See CREATE_BEARER_TOKEN_CREDENTIAL Procedure or CREATE_CREDENTIALS Procedure and Function for further information.

Syntax

```
PROCEDURE UPDATE_BEARER_TOKEN_CREDENTIAL
(
     credential_name           IN VARCHAR2,
     attribute                 IN VARCHAR2,
     new_value                 IN VARCHAR2
);
```

Parameters

Parameter	Description
credential_name	The name of the credential to update.
attribute	The attribute to update. One of 'BEARER_TOKEN', 'CLIENT_ID', 'CLIENT_SECRET', 'TOKEN_REFRESH_RATE'. The token endpoint can not be changed.
new_value	The new value.

Example: Update the `CLIENT_ID` of a credential

```
SQL> BEGIN
  2    dbms_share.create_bearer_token_credential(
  3      credential_name=>'MY_RENEWABLE_CREDENTIAL',
  4      token_endpoint=>'https://myserver/ords/share_provider/oauth/token',
  5      client_id=>'VXGQ_44s6qJ-K4WHUNM2yQ..',
  6      client_secret=>'y9ddppgwEmZl7adDHFQndw..');
  7  END;
  8  /
PL/SQL procedure successfully completed.
SQL> select credential_name, username from user_credentials where
credential_name LIKE 'MY_RENEWABLE_CREDENTIAL%';
CREDENTIAL_NAME
------------------------------------------
USERNAME
----------------------------------
MY_RENEWABLE_CREDENTIAL
BEARER_TOKEN
MY_RENEWABLE_CREDENTIAL$TOKEN_REFRESH_CRED
ABCDEF
```

Summary of Share Producer Views

This table lists the share producer views for the `DBMS_SHARE` package.

View	Description
ALL_OBJECT_SHAREABILITY View	This view gives information about the shareability, or otherwise, of tables and views visible to the user.
ALL_SHAREABLE_OBJECTS View	This view lists all of the tables or views that can be added to a share.
ALL_SHARES View	This view lists all share created using `CREATE_SHARE`.
ALL_SHARE_DEFAULT_SETTINGS View	A view containing the default settings for shares and recipients.
ALL_SHARE_EVENTS View	This view lists all log events associated with shares.
ALL_SHARE_FILES View	A view that contains a list of all the parquet files generated by shares.
ALL_SHARE_RECIPIENTS View	A view containing all share recipients created using `CREATE_SHARE_RECIPIENT`.
ALL_SHARE_RECIPIENT_EVENTS View	This view contains records of endpoint request by delta share consumers.
ALL_SHARE_RECIPIENT_GRANTS View	This view contains one row for each share/recipient combination as specified by calls to `GRANT_TO_RECIPIENT`.
ALL_SHARE_SCHEMAS View	This view lists all the share schemas that were added to shares using `ADD_TO_SHARE`.
ALL_SHARE_TABLES View	This view lists all tables or views that were added to share using `ADD_TO_SHARE`.
ALL_SHARE_VERSIONS View	A view listing the different versions of shares.
DBA_SHARE_DETACHED_FILES View	This view contains a list of parquet files that we generated by `DBMS_SHARE` but that have become detached in some way.

View	Description
DBA_SHARE_EVENTS View	This view contains records of endpoint request by delta share consumers.
DBA_SHARE_FILES View	A view that contains a list of all the parquet files generated by shares.
DBA_SHARE_JOB_SEGMENTS View	This view contains a list of parquet files that we generated by DBMS_SHARE but that have become detached in some way.
DBA_SHARE_JOBS View	List share jobs associated with the current user.
DBA_SHARE_RECIPIENT_EVENTS View	This view contains records of endpoint request by delta share consumers.
DBA_SHARE_RECIPIENT_GRANTS View	This view contains one row for each share/recipient combination as specified by calls to GRANT_TO_RECIPIENT.
DBA_SHARE_RECIPIENTS View	A view containing all share recipients created using CREATE_SHARE_RECIPIENT.
DBA_SHARE_SCHEMAS View	This view lists all the share schemas that were added to shares using ADD_TO_SHARE.
DBA_SHARE_TABLES View	This view lists all tables or views that were added to share using ADD_TO_SHARE.
DBA_SHARE_VERSIONS View	A view listing the different versions of shares.
DBA_SHARES View	This view lists all share created using DBMS_SHARE.CREATE_SHARE.
USER_SHARE_DETACHED_FILES View	This view contains a list of parquet files that we generated by DBMS_SHARE but that have become detached in some way.
USER_SHARE_EVENTS View	This view contains records of endpoint request by delta share consumers.
USER_SHARE_FILES View	A view that contains a list of all the parquet files generated by shares.
USER_SHARE_JOBS View	List share jobs associated with the current user.
USER_SHARE_JOB_SEGMENTS View	List segments (tables, table partitions, or table sub-partitions) that are being actively processed by running share jobs.
USER_SHARE_RECIPIENT_EVENTS View	This view contains records of endpoint request by delta share consumers.
USER_SHARE_RECIPIENT_GRANTS View	This view contains one row for each share/recipient combination as specified by calls to GRANT_TO_RECIPIENT.
USER_SHARE_RECIPIENTS View	A view containing all share recipients created using CREATE_SHARE_RECIPIENT.
USER_SHARE_SCHEMAS View	This view lists all the share schemas that were added to shares using ADD_TO_SHARE.
USER_SHARE_TABLES View	This view lists all tables or views that were added to share using ADD_TO_SHARE.
USER_SHARE_VERSIONS View	A view listing the different versions of shares.
USER_SHAREABLE_OBJECTS View	This view lists all of the tables or views that can be added to a share.
USER_SHARES View	This view lists all share created using DBMS_SHARE.CREATE_SHARE.

- ALL_OBJECT_SHAREABILITY View
  A view containing all share links that were created using `CREATE_SHARE_LINK`.
- ALL_SHAREABLE_OBJECTS View
  This view lists all of the tables or views that can be added to a share. Objects that are not shareable may be listed in `ALL_OBJECT_SHAREABILITY` along with a reason explaining why they are excluded.
- ALL_SHARES View
  This view lists all share created using `DBMS_SHARE.CREATE_SHARE`.
- ALL_SHARE_DEFAULT_SETTINGS View
  A view containing the default settings for shares and recipients.
- ALL_SHARE_EVENTS View
  This view lists all log events associated with shares.
- ALL_SHARE_FILES View
  A view that contains a list of all the parquet files generated by shares.
- ALL_SHARE_RECIPIENTS View
  A view containing all share recipients created using `CREATE_SHARE_RECIPIENT`.
- ALL_SHARE_RECIPIENT_EVENTS View
  This view contains records of endpoint request by delta share consumers.
- ALL_SHARE_RECIPIENT_GRANTS View
  This view contains one row for each share/recipient combination as specified by calls to `GRANT_TO_RECIPIENT`.
- ALL_SHARE_SCHEMAS View
  This view lists all the share schemas that were added to shares using `ADD_TO_SHARE`.
- ALL_SHARE_TABLES View
  This view lists all tables or views that were added to share using `ADD_TO_SHARE`.
- ALL_SHARE_VERSIONS View
  A view listing the different versions of shares. For VERSIONED shares, each row represents a particular set of parquet files. For LIVE shares, each row represents a different list of shared objects.
- DBA_SHARE_DETACHED_FILES View
  This view contains a list of parquet files that were generated by `DBMS_SHARE` but that have become detached in some way.
- DBA_SHARE_EVENTS View
  This view lists all log events associated with shares. Its columns are the same as those in the `ALL_SHARE_EVENTS` view.
- DBA_SHARE_FILES View
  A view that contains a list of all the parquet files generated by shares. Its columns are the same as those in the `ALL_SHARE_FILES` view.
- DBA_SHARE_JOB_SEGMENTS View
  List segments (tables, table partitions, or table sub-partitions) that are being actively processed by running share jobs. Its columns are the same as those in the `USER_SHARE_JOB_SEGMENTS` view.
- DBA_SHARE_JOBS View
  List share jobs associated with the current user. Its columns are the same as those in the `USER_SHARE_JOBS` view with the addition of an `OWNER` column.

- DBA_SHARE_RECIPIENT_EVENTS View
  This view contains records of endpoint request by delta share consumers. Its columns are the same as those in the ALL_SHARE_RECIPIENT_EVENTS view.
- DBA_SHARE_RECIPIENT_GRANTS View
  This view contains one row for each share/recipient combination as specified by calls to GRANT_TO_RECIPIENT. Its columns are the same as those in the ALL_SHARE_RECIPIENT_GRANTS view.
- DBA_SHARE_RECIPIENTS View
  A view containing all share recipients created using CREATE_SHARE_RECIPIENT. Its columns are the same as those in the ALL_SHARE_RECIPIENTS view.
- DBA_SHARE_SCHEMAS View
  This view lists all the share schemas that were added to shares using ADD_TO_SHARE. Its columns are the same as those in the ALL_SHARE_SCHEMAS view.
- DBA_SHARE_TABLES View
  This view lists all tables or views that were added to share using ADD_TO_SHARE. Its columns are the same as those in the ALL_SHARE_TABLES view.
- DBA_SHARE_VERSIONS View
  A view listing the different versions of shares. For VERSIONED shares, each row represents a particular set of parquet files. For LIVE shares, each row represents a different list of shared objects. Its columns are the same as those in the ALL_SHARE_VERSIONS view.
- DBA_SHARES View
  This view lists all share created using DBMS_SHARE.CREATE_SHARE. Its columns are the same as those in the ALL_SHARES view.
- USER_SHARE_DETACHED_FILES View
  This view contains a list of parquet files that were generated by DBMS_SHARE but that have become detached in some way.
- USER_SHARE_EVENTS View
  This view lists all log events associated with shares. Its columns are the same as those in the ALL_SHARE_EVENTS view.
- USER_SHARE_FILES View
  A view that contains a list of all the parquet files generated by shares. Its columns are the same as those in the ALL_SHARE_FILES view.
- USER_SHARE_JOBS View
  List share jobs associated with the current user.
- USER_SHARE_JOB_SEGMENTS View
  List segments (tables, table partitions, or table sub-partitions) that are being actively processed by running share jobs.
- USER_SHARE_RECIPIENT_EVENTS View
  This view contains records of endpoint request by delta share consumers. Its columns are the same as those in the ALL_SHARE_RECIPIENT_EVENTS view.
- USER_SHARE_RECIPIENT_GRANTS View
  This view contains one row for each share/recipient combination as specified by calls to GRANT_TO_RECIPIENT. Its columns are the same as those in the ALL_SHARE_RECIPIENT_GRANTS view.
- USER_SHARE_RECIPIENTS View
  A view containing all share recipients created using CREATE_SHARE_RECIPIENT. Its columns are the same as those in the ALL_SHARE_RECIPIENTS view.

- USER_SHARE_SCHEMAS View
  This view lists all the share schemas that were added to shares using ADD_TO_SHARE. Its columns are the same as those in the ALL_SHARE_SCHEMAS view.
- USER_SHARE_TABLES View
  This view lists all tables or views that were added to share using ADD_TO_SHARE. Its columns are the same as those in the ALL_SHARE_SCHEMAS view.
- USER_SHARE_VERSIONS View
  A view listing the different versions of shares. For VERSIONED shares, each row represents a particular set of parquet files. For LIVE shares, each row represents a different list of shared objects. Its columns are the same as those in the ALL_SHARE_VERSIONS view.
- USER_SHAREABLE_OBJECTS View
  This view lists all of the tables or views that can be added to a share. Objects that are not shareable may be listed in ALL_OBJECT_SHAREABILITY along with a reason explaining why they are excluded. Its columns are the same as those in the ALL_SHAREABLE_OBJECTS view.
- USER_SHARES View
  This view lists all share created using DBMS_SHARE.CREATE_SHARE.

ALL_OBJECT_SHAREABILITY View

A view containing all share links that were created using CREATE_SHARE_LINK.

See CREATE_SHARE_LINK Procedure for further information.

Column	Datatype	Description
OWNER	VARCHAR2	The owner of the object.
OBJECT_NAME	VARCHAR2	The name of the object.
OBJECT_TYPE	VARCHAR2	The type of the object, TABLE or VIEW.
NUM_ROWS	NUMBER	The number of rows, if this is a TABLE and if the statistics are available.
PRIV_CODE	VARCHAR2	A non NULL value, if you do not have the required privileges to share this object. • NO_GRANT_READ_PRIV: You have READ privilege, but not READ WITH GRANT OPTION. • PRIV_ERROR: This is an Oracle-maintained object or is otherwise unsupported
SUPPORT_CODE	VARCHAR2	A non NULL value, if this object cannot be shared. • INVALID: The object is INVALID (e.g. a badly defined VIEW). • SHARDED: The table is sharded. • NESTED: This is a nested table. • EXTERNAL_PARTITIONED: This is an partitioned, external table. • EXTERNAL: This is a non-partitioned external table. • INDEX_ORGANIZED_TABLE: This is an index organized table. • TRACKING_MV: This is a tracking MV created by DBMS_SHARE. • NO_SHAREABLE_COLUMNS: None of the columns in the table or view can be shared.

Column	Datatype	Description
SHARE_METADATA	BLOB	A JSON document that gives details of which columns are shareable and how they would be represented using parquet.

ALL_SHAREABLE_OBJECTS View

This view lists all of the tables or views that can be added to a share. Objects that are not shareable may be listed in ALL_OBJECT_SHAREABILITY along with a reason explaining why they are excluded.

Column	Datatype	Description
OWNER	VARCHAR2	The owner of the object.
OBJECT_NAME	VARCHAR2	The name of the object.
OBJECT_TYPE	VARCHAR2	The type of the object, TABLE or VIEW.
NUM_ROWS	NUMBER	The number of rows, if this is a TABLE and if the statistics are available.
SHARE_METADATA	BLOB	A JSON document that gives details of which columns are shareable and how they would be represented using parquet.
NUM_COLUMNS	VARCHAR2	The number of shareable columns in the object.

ALL_SHARES View

This view lists all share created using DBMS_SHARE.CREATE_SHARE.

See CREATE_SHARE Procedure for further information.

Column	Datatype	Description
OWNER	VARCHAR2	The owner of the share.
SHARE_NAME	VARCHAR2	The name of the share.
SHARE_ID	NUMBER	A numeric ID for this share.
METADATA_PATH	VARCHAR2	The lineage metadata path for this share.
METADATA	BLOB	A JSON document containing additional share metadata.
DESCRIPTION	CLOB	A local description of the share.
SHARE_TYPE	VARCHAR2	The type is either VERSIONED or LIVE.
PUBLIC_DESCRIPTION	CLOB	An externally visible description of the share.
NUM_TABLES	NUMBER	The number of tables in the share.
NUM_RECIPIENTS	NUMBER	The number of recipients who have been granted access to the share.
NUM_ORACLE_RECIPIENTS	NUMBER	The number of Oracle recipients who have been granted access to the share.
STORAGE_FOLDER	VARCHAR2	The base URL for any generated parquet files.
ORACLE_SHARE_ID	VARCHAR2	A globally unique UUID for the share.
LAST_EXPORT_TIME	TIMESTAMP(9) WITH TIME ZONE	(VERSIONED ONLY) The time when the currently published version was generated.
LAST_NOTIFICATION	TIMESTAMP(9) WITH TIME ZONE	The time the last email notification was sent about this share..

Column	Datatype	Description
CURRENT_VERSION	NUMBER	The currently published version number.
CREATED	TIMESTAMP(9) WITH TIME ZONE	The date this share was created.
UPDATED	TIMESTAMP(9) WITH TIME ZONE	The date this share was last modified.

ALL_SHARE_DEFAULT_SETTINGS View

A view containing the default settings for shares and recipients.

These can be modified (by ADMIN) using `UPDATE_DEFAULT_SHARE_PROPERTY` and `UPDATE_DEFAULT_RECIPIENT_PROPERTY`. Note that there is no DBA or USER version of this view.

Column	Datatype	Description
SETTING	VARCHAR2	The setting name: • JOB_CLASS • JOB_DOP • MAX_PAR_LIFETIME • MAX_TOKEN_LIFETIME • MIN_PAR_LIFETIME • PAR_LIFETIME • SPLIT_SIZE • TOKEN_LIFETIME
SETTING_VALUE	VARCHAR2	The current setting value.

ALL_SHARE_EVENTS View

This view lists all log events associated with shares.

Entries can be removed by deleting the associated share or by calling the CLEAR_SHARE_EVENTS Procedure.

Column	Datatype	Description
OWNER	VARCHAR2	The owner of the share.
SHARE_NAME	VARCHAR2	The name of the share.
SHARE_ID	NUMBER	A numeric ID for the share.
STATUS	VARCHAR2	An event status, such as INFO or ERROR.
DETAILS	CLOB	Additional data for the event, typically a JSON document.
TIME	TIMESTAMP(6) WITH TIME ZONE	The time of the event.
LOG_LEVEL	NUMBER	The detail level for the event: 1-3
SHARE_JOB_ID	NUMBER	The ID of the share job (from USER_SHARE_JOBS View) that created this event.
SCHEDULER_JOB_NAME	VARCHAR2	The name of the `dbms_scheduler` job that created this event.
ACTION	VARCHAR2	A code for the event activity.

Column	Datatype	Description
SHARE_VERSION	NUMBER	The share version number associated with this event, if relevant.
OBJECT_NAME	VARCHAR2	The name of the table or view being processed, if relevant.
OBJECT_OWNER	VARCHAR2	The owner of the table or view name being processed, if relevant.
PARTITION_NAME	VARCHAR2	The partition of the table being processed, if relevant.
SPLIT_ORDER	NUMBER	The split number, if a single partition is broken into splits.
SQL_STATEMENT	CLOB	A SQL statement that was executed as part of the event.

ALL_SHARE_FILES View

A view that contains a list of all the parquet files generated by shares.

Column	Datatype	Description
OWNER	VARCHAR2	The owner of the share.
SHARE_NAME	VARCHAR2	The name of the share.
SHARE_ID	NUMBER	A numeric ID for the share.
SHARE_TABLE_NAME	VARCHAR2	The share table associated with the file.
SHARE_SCHEMA_NAME	VARCHAR2	The share schema associated with the file.
SHARE_TABLE_ID	NUMBER	The numeric ID of the share table (ALL_SHARE_TABLES).
SEGMENT_NAME	VARCHAR2	The associated segment name (e.g. a partition), if appropriate.
INITIAL_SHARE_VERSION	NUMBER	The lowest share version number that requires this file.
EXPIRED_SHARE_VERSION	NUMBER	The share version number, if any, where this file was removed.
FLE_URI	VARCHAR2	The full URI for the file.
FILE_NAME	VARCHAR2	The name of the file within the cloud storage bucket.
BYTES	NUMBER	The number of bytes in the file.
SPLIT_ORDER	NUMBER	The split number within the segment.
FILE_STATS	BLOB	Delta statistics (min/max) for columns in the file.

ALL_SHARE_RECIPIENTS View

A view containing all share recipients created using `CREATE_SHARE_RECIPIENT`.

See CREATE_SHARE_RECIPIENT Procedure for further information.

Name	Datatype	Description
OWNER	VARCHAR2	The owner of the recipient object.
RECIPIENT_NAME	VARCHAR2	The name of the recipient.
RECIPIENT_ID	NUMBER	A numeric ID for the recipient
METADATA_PATH	VARCHAR2	The lineage path for the recipient.

Name	Datatype	Description
RECIPIENT_TYPE	VARCHAR2	The type of recipient • DELTA_SHARING: recipient defined by an email • ORACLE: defined by an Oracle sharing ID • ORACLE_DELTA: defined by an email and a sharing ID
INFO	BLOB	Not currently used.
METADATA	BLOB	JSON metadata associated with this recipient.
DESCRIPTION	CLOB	A local description of the recipient.
RECIPIENT_SHARING_ID	VARCHAR2	(ORACLE) An optional sharing ID for the recipient.
MIN_PAR_INTERVAL	INTERVAL DAY(3) TO SECOND(0)	(DELTA) The shortest lifetime of pre-authenticated requests that will be sent to the consumer.
MAX_PAR_INTERVAL	INTERVAL DAY(3) TO SECOND(0)	(DELTA) The longest lifetime of pre-authenticated requests that will be sent to the consumer.
LOG_LEVEL	NUMBER	The recipient event log level: 0 - 3
CREATED	TIMESTAMP(6) WITH TIME ZONE	The date this share recipient was created.
UPDATED	TIMESTAMP(6) WITH TIME ZONE	The date this share recipient was last modified.

ALL_SHARE_RECIPIENT_EVENTS View

This view contains records of endpoint request by delta share consumers.

Name	Type	Description
OWNER	VARCHAR2	The owner of the recipient.
RECIPIENT_NAME	VARCHAR2	The name of the recipient.
RECIPIENT_ID	NUMBER	The numeric ID of the recipient.
SHARE_ID	NUMBER	The ID of the share being accessed.
STATUS	VARCHAR2	A status code, such as INFO or ERROR.
ENDPOINT	VARCHAR2	The delta endpoint being accessed.
REMOTE_ADDRESS	VARCHAR2	The external address that called the endpoint.
HTTP_USER_AGENT	VARCHAR2	The requesting user-agent, from the HTTP request.
HTTP_REFERER	VARCHAR2	The referer from the HTTP request.
REQUEST	CLOB	The request body. This is only logged if the recipient event level is high.
ERROR_MESSAGE	VARCHAR2	Any error message, if the request failed.
ERROR_TRACE	VARCHAR2	Any error trace, if the request failed.
ELAPSED	INTERVAL DAY(9) TO SECOND(9)	The elapsed time taken to process the request.
CPU_TIME	NUMBER	The CPU time take to process the request
DETAILS	CLOB	The response details. This is only logged if the recipient event level is high.
TIME	TIMESTAMP(6) WITH TIME ZONE	The time the event was recorded.
LOG_LEVEL	NUMBER	The detail level of this event: 1-3

ALL_SHARE_RECIPIENT_GRANTS View

This view contains one row for each share/recipient combination as specified by calls to `GRANT_TO_RECIPIENT`.

See GRANT_TO_RECIPIENT Procedure for further information.

Name	Datatype	Description
OWNER	VARCHAR2	The owner of the share and recipient.
RECIPIENT_NAME	VARCHAR2	The name of the recipient.
RECIPIENT_ID	NUMBER	The numeric ID of the recipient.
SHARE_NAME	VARCHAR2	The name of the share.
SHARE_ID	NUMBER	The numeric ID of the share.
ORDER_NUM	NUMBER	The grant order.
CREATED	TIMESTAMP(6) WITH TIME ZONE	The time when the grant was made.
UPDATED	TIMESTAMP(6) WITH TIME ZONE	The last time when this grant was updated.

ALL_SHARE_SCHEMAS View

This view lists all the share schemas that were added to shares using `ADD_TO_SHARE`.

See ADD_TO_SHARE Procedure for further information.

Name	Datatype	Description
OWNER	VARCHAR2	The owner of the share.
SHARE_NAME	VARCHAR2	The name of the share.
SHARE_ID	NUMBER	The numeric ID for this share.
SCHEMA_NAME	VARCHAR2	The name of the share schema, as seen by the share consumers.
SCHEMA_ID	NUMBER	The numeric ID of the share schema.
METADATA_PATH	NUMBER	The lineage path for this share schema.
CREATED	TIMESTAMP(6) WITH TIME ZONE	The date this share schema was created.
UPDATED	TIMESTAMP(6) WITH TIME ZONE	The date this share schema was last modified.

ALL_SHARE_TABLES View

This view lists all tables or views that were added to share using `ADD_TO_SHARE`.

See ADD_TO_SHARE Procedure for further information.

Name	Datatype	Description
OWNER	VARCHAR2	The owner of the share.
SHARE_NAME	VARCHAR2	The name of the share.
SHARE_ID	NUMBER	A numeric ID for the share.

Name	Datatype	Description
SHARE_SCHEMA_NAME	VARCHAR2	The name of the share schema, as seen by the share consumers.
SHARE_SCHEMA_ID	NUMBER	A numeric ID for the share schema.
SHARE_TABLE_NAME	VARCHAR2	The name of the share schema, as seen by the share consumers.
SHARE_TABLE_ID	NUMBER	A numeric ID for the share table.
METADATA_PATH	VARCHAR2	The lineage path for the share table.
TABLE_NAME	VARCHAR2	The local table or view being shared.
TABLE_OWNER	VARCHAR2	The owner of the local table or view being shared.
OBJECT_TYPE	VARCHAR2	The type of object being shared: TABLE or VIEW.
CREATED	TIMESTAMP(6) WITH TIME ZONE	The date this share table was created.
UPDATED	TIMESTAMP(6) WITH TIME ZONE	The date this share table was last modified.
STATUS	VARCHAR2	The STATUS of the shared object, from ALL_OBJECTS.
PRIV_CODE	VARCHAR2	See ALL_OBJECT_SHAREABILITY.PRIV_CODE.

ALL_SHARE_VERSIONS View

A view listing the different versions of shares. For VERSIONED shares, each row represents a particular set of parquet files. For LIVE shares, each row represents a different list of shared objects.

Name	Datatype	Description
OWNER	VARCHAR2	The owner of the share.
SHARE_NAME	VARCHAR2	The name of the share.
SHARE_ID	NUMBER	A numeric ID for the share.
SHARE_VERSION	NUMBER	The version of this share.
STATUS	VARCHAR2	The status of this version. • CURRENT • RETIRED • FAILED • ROLLED BACK • STOPPED • PENDING • EXPORTING
EXPORT_TIME	TIMESTAMP(9)	The time when this version was created.
LAST_UPDATE	TIMESTAMP(9)	The last time this version was modified.
PROMISED_TO	TIMESTAMP(9)	The latest expiry time for any pre-authenticated request on a parquet file associated with this version.
EXPORT_SCN	NUMBER	The System Change Number current when this version was created.
NUM_FILES	NUMBER	The number of parquet files associated with this version.
NUM_NEW_FILES	NUMBER	The number of new parquet files created for this version.
NUM_PENDING_FILES	NUMBER	The number of parquet files that should have been created, but were not for some reason.
NUM_NEW_PENDING_FILES	NUMBER	The number of pending files that are new as of this version.

Name	Datatype	Description
ERROR_MESSAGE	VARCHAR2	An error message in the case where the version failed.
TOTAL_FILE_SIZE	NUMBER	The total size of all parquet files associated with this version.

DBA_SHARE_DETACHED_FILES View

This view contains a list of parquet files that were generated by DBMS_SHARE but that have become detached in some way.

This could happen, for example, if a share or share version is dropped with the destroy_objects argument set to FALSE. It can also happen if a user with shares is dropped.

Name	Datatype	Description
OWNER	VARCHAR2	The name of the user who generated the file. This may be NULL if the user has been dropped from the database.
OWNER_ID	NUMBER	The USER_ID of the user who generated the file. This will remain valid and non-NULL even if the user was dropped.
FILE_URI	VARCHAR2	The URL of the generated parquet file.

See Also:

"USER_SHARE_DETACHED_FILES View"

DBA_SHARE_EVENTS View

This view lists all log events associated with shares. Its columns are the same as those in the ALL_SHARE_EVENTS view.

See Also:

"ALL_SHARE_EVENTS View"

DBA_SHARE_FILES View

A view that contains a list of all the parquet files generated by shares. Its columns are the same as those in the ALL_SHARE_FILES view.

See Also:

"ALL_SHARE_FILES View"

DBA_SHARE_JOB_SEGMENTS View

List segments (tables, table partitions, or table sub-partitions) that are being actively processed by running share jobs. Its columns are the same as those in the USER_SHARE_JOB_SEGMENTS view.

See Also:

"USER_SHARE_JOB_SEGMENTS View"

DBA_SHARE_JOBS View

List share jobs associated with the current user. Its columns are the same as those in the `USER_SHARE_JOBS` view with the addition of an `OWNER` column.

See Also:

"USER_SHARE_JOBS View"

DBA_SHARE_RECIPIENT_EVENTS View

This view contains records of endpoint request by delta share consumers. Its columns are the same as those in the `ALL_SHARE_RECIPIENT_EVENTS` view.

See Also:

"ALL_SHARE_RECIPIENT_EVENTS View"

DBA_SHARE_RECIPIENT_GRANTS View

This view contains one row for each share/recipient combination as specified by calls to `GRANT_TO_RECIPIENT`. Its columns are the same as those in the `ALL_SHARE_RECIPIENT_GRANTS` view.

See Also:

"ALL_SHARE_RECIPIENT_GRANTS View"

DBA_SHARE_RECIPIENTS View

A view containing all share recipients created using `CREATE_SHARE_RECIPIENT`. Its columns are the same as those in the `ALL_SHARE_RECIPIENTS` view.

See Also:

" ALL_SHARE_RECIPIENTS View"

DBA_SHARE_SCHEMAS View

This view lists all the share schemas that were added to shares using ADD_TO_SHARE. Its columns are the same as those in the `ALL_SHARE_SCHEMAS` view.

See Also:

"ALL_SHARE_SCHEMAS View"

DBA_SHARE_TABLES View

This view lists all tables or views that were added to share using `ADD_TO_SHARE`. Its columns are the same as those in the `ALL_SHARE_TABLES` view.

See Also:

"ALL_SHARE_TABLES View"

DBA_SHARE_VERSIONS View

A view listing the different versions of shares. For VERSIONED shares, each row represents a particular set of parquet files. For LIVE shares, each row represents a different list of shared objects. Its columns are the same as those in the `ALL_SHARE_VERSIONS` view.

See Also:

"ALL_SHARE_VERSIONS View"

DBA_SHARES View

This view lists all share created using `DBMS_SHARE.CREATE_SHARE`. Its columns are the same as those in the `ALL_SHARES` view.

See Also:

"ALL_SHARES View"

USER_SHARE_DETACHED_FILES View

This view contains a list of parquet files that were generated by `DBMS_SHARE` but that have become detached in some way.

This could happen, for example, if a share or share version is dropped with the `destroy_objects` argument set to FALSE. It can also happen if a user with shares is dropped. In that case the files would show up in `DBA_SHARE_DETACHED_FILES View`.

Name	Datatype	Description
`FILE_URI`	`VARCHAR2`	The URL of the generated parquet file.

See Also:

"DBA_SHARE_DETACHED_FILES View"

USER_SHARE_EVENTS View

This view lists all log events associated with shares. Its columns are the same as those in the `ALL_SHARE_EVENTS` view.

See Also:

"ALL_SHARE_EVENTS View"

USER_SHARE_FILES View

A view that contains a list of all the parquet files generated by shares. Its columns are the same as those in the `ALL_SHARE_FILES` view.

See Also:

"ALL_SHARE_FILES View"

USER_SHARE_JOBS View

List share jobs associated with the current user.

Share jobs are created whenever the database needs to run share processes in the background. The PUBLISH_SHARE Procedure initiates a share job.

Column	Datatype	Description
JOB_ID	NUMBER	A numeric ID for the job.
JOB_NAME	VARCHAR2	The name of the share job.
SHARE_ID	NUMBER	A numeric ID for the share associated with this job, if relevant.
PARALLELISM	NUMBER	The number of DBMS_SCHEDULER jobs used to run this share job.
SHARE_NAME	VARCHAR2	The name of the share associated with this job, if relevant.
SHARE_VERSION	NUMBER	The share version number associated with this job, if relevant.
STATUS	VARCHAR2	The status of this job: • PENDING • STOPPING • RUNNING • WAITING_FOR_ODI • DELETING • FAILED • GENERATING_PARS • COMPLETED • STOPPED • ERROR

USER_SHARE_JOB_SEGMENTS View

List segments (tables, table partitions, or table sub-partitions) that are being actively processed by running share jobs.

Name	Datatype	Description
JOB_ID	NUMBER	A numeric ID for the job from USER_SHARE_JOBS.
JOB_NAME	VARCHAR2	The name of the share job from USER_SHARE_JOBS.
SHARE_ID	NUMBER	A numeric ID for the share associated with this job, if relevant
PARALLELISM	NUMBER	The number of DBMS_SCHEDULER jobs used to run this share job.
SHARE_NAME	VARCHAR2	The name of the share associated with this job, if relevant.
SHARE_VERSION	NUMBER	The share version number associated with this job, if relevant.

Name	Datatype	Description
STATUS	VARCHAR2	The status of this job: • PENDING • STOPPING • RUNNING • WAITING_FOR_ODI • DELETING • FAILED • GENERATING_PARS • COMPLETED • STOPPED • ERROR
OBJECT_NAME	VARCHAR2	The name of the object that is being processed.
OBJECT_OWNER	VARCHAR2	The owner of the object that is being processed.
OBJECT_TYPE	VARCHAR2	The type of object (TABLE or VIEW) that is being processed.
SHARE_TABLE_NAME	VARCHAR2	The name of the object as it appears in the share.
SHARE_SCHEMA_NAME	VARCHAR2	The name of the object's schema as it appears in the share.
PARTITION_NAME	VARCHAR2	The name of the active partition or sub-partition, if appropriate.
SEGMENT_TYPE	VARCHAR2	The type of segment: TABLE, TABLE PARTITION, or TABLE SUBPARTITION.
BYTES	VARCHAR2	The size of the segment in bytes.

See Also:

"DBA_SHARE_JOB_SEGMENTS View"

USER_SHARE_RECIPIENT_EVENTS View

This view contains records of endpoint request by delta share consumers. Its columns are the same as those in the ALL_SHARE_RECIPIENT_EVENTS view.

See Also:

"ALL_SHARE_RECIPIENT_EVENTS View"

USER_SHARE_RECIPIENT_GRANTS View

This view contains one row for each share/recipient combination as specified by calls to GRANT_TO_RECIPIENT. Its columns are the same as those in the ALL_SHARE_RECIPIENT_GRANTS view.

See Also:

"ALL_SHARE_RECIPIENT_GRANTS View"

USER_SHARE_RECIPIENTS View

A view containing all share recipients created using `CREATE_SHARE_RECIPIENT`. Its columns are the same as those in the `ALL_SHARE_RECIPIENTS` view.

See Also:

" ALL_SHARE_RECIPIENTS View"

USER_SHARE_SCHEMAS View

This view lists all the share schemas that were added to shares using `ADD_TO_SHARE`. Its columns are the same as those in the `ALL_SHARE_SCHEMAS` view.

See Also:

"ALL_SHARE_SCHEMAS View"

USER_SHARE_TABLES View

This view lists all tables or views that were added to share using `ADD_TO_SHARE`. Its columns are the same as those in the `ALL_SHARE_SCHEMAS` view.

See Also:

"ALL_SHARE_TABLES View"

USER_SHARE_VERSIONS View

A view listing the different versions of shares. For VERSIONED shares, each row represents a particular set of parquet files. For LIVE shares, each row represents a different list of shared objects. Its columns are the same as those in the `ALL_SHARE_VERSIONS` view.

See Also:

"ALL_SHARE_VERSIONS View"

USER_SHAREABLE_OBJECTS View

This view lists all of the tables or views that can be added to a share. Objects that are not shareable may be listed in `ALL_OBJECT_SHAREABILITY` along with a reason explaining why they are excluded. Its columns are the same as those in the `ALL_SHAREABLE_OBJECTS` view.

See Also:

"ALL_SHAREABLE_OBJECTS View"

USER_SHARES View

This view lists all share created using `DBMS_SHARE.CREATE_SHARE`.

See Also:

"ALL_SHARES View"

Summary of Share Consumer Views

This table lists the DBMS_SHARE package views.

View	Description
ALL_AVAILABLE_ORACLE_SHARE_PROVIDERS View	This view lists all available Oracle share providers.
ALL_SHARE_CACHE_JOB_INFO View	This view contains information on the most recent Oracle Live share cache population job in the current POD.
ALL_SHARE_LINKS View	A view containing all share links that were created using CREATE_SHARE_LINK.
ALL_SHARE_LINK_VIEWS View	A view that contains a list of all the views created to access the shared data using CREATE_SHARE_LINK_VIEW.
ALL_SHARE_PROVIDERS View	A view that contains one row for each share provider subscription, created by CREATE_SHARE_PROVIDER.
DBA_SHARE_LINK_VIEWS View	A view that contains a list of all the views created to access the shared data using CREATE_SHARE_LINK_VIEW.
DBA_SHARE_LINKS View	A view containing all share links that were created using CREATE_SHARE_LINK.
USER_SHARE_LINKS_VIEW View	A view that contains a list of all the views created to access the shared data using CREATE_SHARE_LINK_VIEW.
USER_SHARE_LINKS View	A view containing all share links that were created using CREATE_SHARE_LINK.
USER_SHARE_PROVIDERS View	A view that contains one row for each share provider subscription, created by CREATE_SHARE_PROVIDER.

- ALL_AVAILABLE_ORACLE_SHARE_PROVIDERS View
  This view lists all available Oracle share providers.
- ALL_SHARE_CACHE_JOB_INFO View
  This view contains information on the most recent Oracle Live share cache population job in the database.
- ALL_SHARE_LINK_VIEWS View
  A view that contains a list of all the views created to access the shared data using CREATE_SHARE_LINK_VIEW.
- ALL_SHARE_LINKS View
  A view containing all share links that were created using CREATE_SHARE_LINK.
- ALL_SHARE_PROVIDERS View
  A view that contains one row for each share provider subscription, created by CREATE_SHARE_PROVIDER.
- DBA_SHARE_LINK_VIEWS View
  A view that contains a list of all the views created to access the shared data using CREATE_SHARE_LINK_VIEW. Its columns are the same as those in the ALL_SHARE_LINK_VIEWS view.

- DBA_SHARE_LINKS View
  A view containing all share links that were created using `CREATE_SHARE_LINK`. Its columns are the same as those in the `ALL_SHARE_LINKS` view.
- USER_SHARE_LINKS_VIEW View
  A view that contains a list of all the views created to access the shared data using `CREATE_SHARE_LINK_VIEW`. Its columns are the same as those in the `ALL_SHARE_LINK_VIEWS` view.
- USER_SHARE_LINKS View
  A view containing all share links that were created using `CREATE_SHARE_LINK`. Its columns are the same as those in the `ALL_SHARE_LINKS` view.
- USER_SHARE_PROVIDERS View
  A view that contains one row for each share provider subscription, created by `CREATE_SHARE_PROVIDER`. Its columns are the same as those in the `ALL_SHARE_PROVIDERS` view.

ALL_AVAILABLE_ORACLE_SHARE_PROVIDERS View

This view lists all available Oracle share providers.

Column	Datatype	Description
ORACLE_PROVIDER_ID	VARCHAR2	A UUID for the Oracle provider.
PROVIDER_NAME	VARCHAR2	A name, as specified by the provider. Note that this is not unique.
DESCRIPTION	VARCHAR2	A description, as specified by the provider.
CONTACT	VARCHAR2	A contact address, such as an email, for the provider.
METADATA	CLOB	Additional JSON metadata, as specified by the provider.
SHARES	CLOB	A JSON representation of the list of shares available from this provider.
CREATED	TIMESTAMP(6)	The time when the share provider was first registered.
UPDATED	TIMESTAMP(6)	The time when the share provider was last updated.

ALL_SHARE_CACHE_JOB_INFO View

This view contains information on the most recent Oracle Live share cache population job in the database.

Names	Datatype	Description
LAST_RUN_DATE	TIMESTAMP(6) WITH TIME ZONE	The start time of the most recent cache job execution.
NEXT_RUN_DATE	TIMESTAMP(6) WITH TIME ZONE	The scheduled start time of the next cache job execution.
RUN_STATUS	NUMBER	The status of the most recent run (0 = success). A non zero value represents the numeric code of the exception, as reported by the SQL CODE function.
MESSAGE_TEXT	VARCHAR2(4000)	A text indicating a warning or error condition encountered by the most recent run. NULL in most cases; indicates a warning if `RUN_STATUS` is 0, an error otherwise.
REPEAT_INTERVAL	VARCHAR2(4000)	The `DBMS_SHEDULER` repeat interval for the cache job.

ALL_SHARE_LINK_VIEWS View

A view that contains a list of all the views created to access the shared data using `CREATE_SHARE_LINK_VIEW`.

See CREATE_SHARE_LINK_VIEW Procedure for further information.

Names	Datatype	Description
LINK_OWNER	VARCHAR2	The owner of the associated share link.
LINK_NAME	VARCHAR2	The name of the associated share link.
SHARE_SCHEMA_NAME	VARCHAR2	The name of the share schema, as supplied by the share provider.
SHARE_TABLE_NAME	VARCHAR2	The name of the share table, as supplied by the share provider.
VIEW_OWNER	VARCHAR2	The owner of the local view.
VIEW_NAME	VARCHAR2	The name of the local view.

ALL_SHARE_LINKS View

A view containing all share links that were created using `CREATE_SHARE_LINK`.

See CREATE_SHARE_LINK Procedure for further information.

Column	Datatype	Description
OWNER	VARCHAR2	The owner of the share link.
LINK_NAME	VARCHAR2	The name of the share link.
LINK_ID	NUMBER	A numeric ID for the share link.
SHARE_TYPE	VARCHAR2	The type of share provider: DELTA or ORACLE.
SHARE_NAME	VARCHAR2	The name of the share.
SHARE_PROVIDER_NAME	VARCHAR2	The name of the share provider.
SHARE_PROVIDER_OWNER	VARCHAR2	The owner of the share provider.
SHARE_PROVIDER_ID	NUMBER	A numeric ID for the provider.
METADATA_PATH	VARCHAR2	The lineage path of the share link.
SHARE_CREDENTIAL_NAME	VARCHAR2	(DELTA) The name of the credential used to access the endpoints.
METADATA	BLOB	Additional metadata, supplied by the user.
ORACLE_SHARE_PROVIDER_ID	VARCHAR2	(ORACLE) A globally unique GUID for the Oracle share provider.
CREATED	TIMESTAMP(6) WITH TIME ZONE	The date this share link was created.
UPDATED	TIMESTAMP(6) WITH TIME ZONE	The date this share link was last modified.

ALL_SHARE_PROVIDERS View

A view that contains one row for each share provider subscription, created by `CREATE_SHARE_PROVIDER`.

See CREATE_SHARE_PROVIDER Procedure for further information.

Column	Datatype	Description
OWNER	VARCHAR2	The user who subscribed to the share provider.
PROVIDER_NAME	VARCHAR2	The local name given to the share provider.
PROVIDER_ID	VARCHAR2	A numeric ID for the provider.
ENDPOINT	VARCHAR2	(DELTA) The delta endpoint for the provider.
SHARE_TYPE	VARCHAR2	The type of share provider: DELTA or ORACLE.
METADATA_PATH	VARCHAR2	The lineage path for the share provider.
SHARE_CREDENTIAL_NAME	VARCHAR2	(DELTA) The name of the credential used to access the delta endpoints.
METADATA	BLOB	Additional metadata, provided by the user.
ORACLE_PROVIDER_ID	VARCHAR2	(ORACLE) A globally unique GUID for the Oracle share provider.
CREATED	TIMESTAMP(6) WITH TIME ZONE	The date this share provider was created.
UPDATED	TIMESTAMP(6) WITH TIME ZONE	The date this share provider was last modified.

DBA_SHARE_LINK_VIEWS View

A view that contains a list of all the views created to access the shared data using CREATE_SHARE_LINK_VIEW. Its columns are the same as those in the ALL_SHARE_LINK_VIEWS view.

See Also:

"ALL_SHARE_LINK_VIEWS View"

DBA_SHARE_LINKS View

A view containing all share links that were created using CREATE_SHARE_LINK. Its columns are the same as those in the ALL_SHARE_LINKS view.

See Also:

"ALL_SHARE_LINKS View"

USER_SHARE_LINKS_VIEW View

A view that contains a list of all the views created to access the shared data using CREATE_SHARE_LINK_VIEW. Its columns are the same as those in the ALL_SHARE_LINK_VIEWS view.

See Also:

"ALL_SHARE_LINK_VIEWS View"

USER_SHARE_LINKS View

A view containing all share links that were created using `CREATE_SHARE_LINK`. Its columns are the same as those in the `ALL_SHARE_LINKS` view.

See Also:

"ALL_SHARE_LINKS View"

USER_SHARE_PROVIDERS View

A view that contains one row for each share provider subscription, created by `CREATE_SHARE_PROVIDER`. Its columns are the same as those in the `ALL_SHARE_PROVIDERS` view.

See Also:

"ALL_SHARE_PROVIDERS View"

DBMS_SHARE Constants

These constants are used by the `DBMS_SHARE` package.

Detached Files

Constants used by PURGE_DETACHED_FILES Procedure.

Name	Type	Value	Description
`PURGE_DROP`	`PLS_INTEGER`	1	Attempt to drop the files, using the specified credential. If the file cannot be dropped, then the file continues to be listed in the `*_SHARE_DETACHED_FILES` views.
`PURGE_DROP_FORCE`	`PLS_INTEGER`	2	Attempt to drop the files, using the specified credential. The file is removed from the `*_SHARE_DETACHED_FILES` views, even if the attempt to drop the file fails again.
`PURGE_FORGET`	`PLS_INTEGER`	3	Remove the files `*_SHARE_DETACHED_FILES` views without attempting to drop them.

Log Level

Constants used for setting the level of information collected in the log. These levels control the number of events that get logged into `ALL_SHARE_EVENTS` and `ALL_SHARE_RECIPIENTS`.

See SET_SHARE_LOG_LEVEL Procedure and SET_RECIPIENT_LOG_LEVEL Procedure.

Name	Type	Value	Description
`LOG_LEVEL_ERRORS_ONLY`	`PLS_INTEGER`	0	Only log errors.

Name	Type	Value	Description
LOG_LEVEL_BASIC	PLS_INTEGER	1	Log errors and basic information.
LOG_LEVEL_DETAIL	PLS_INTEGER	2	Log errors and additional details.
LOG_LEVEL_DEBUG	PLS_INTEGER	3	Log debug level of information.

Share Job Properties

Properties used in UPDATE_SHARE_JOB_PROPERTY Procedure.

Name	Type	Value	Description
PROP_SHARE_JOB_DOP	VARCHAR2(7)	'JOB_DOP'	The JOB_DOP property determines how many DBMS_SCHEDULER jobs are used to publish the share. This should be a number between 1 and 5.
PROP_SHARE_JOB_CLASS	VARCHAR2(9)	'JOB_CLASS'	The JOB_CLASS property determines the scheduler job class that is used to published the share. See ALL_SCHEDULER_JOB_CLASSES for a list of valid values. The default value is DEFAULT_JOB_CLASS.
PROP_SHARE_JOB_PRIORITY	VARCHAR2(12)	'JOB_PRIORITY'	The JOB_PRIORITY property determines the relative priority when two or more shares from the same user are being published at the same time. The value should be a number, with 0 as the default. Shares with a higher priority will be processed before shares with a lower priority. Shares with the same priority are processed on a first come, first serve basis. Standard values for job priority: SHARE_PRIORITY_LOW CONSTANT NUMBER := 1; SHARE_PRIORITY_DEFAULT CONSTANT NUMBER := 2; SHARE_PRIORITY_HIGH CONSTANT NUMBER := 3;

Share Properties

Constants used to indicate the properties of a share.

See UPDATE_SHARE_PROPERTY Procedure, UPDATE_DEFAULT_SHARE_PROPERTY Procedure and GET_SHARE_PROPERTY Function.

Name	Type	Value	Description
PROP_SHARE_DESC	VARCHAR2(11)	'DESCRIPTION'	A locally visible description of the share.
PROP_SHARE_PUBLIC_DESC	VARCHAR2(18)	'PUBLIC_DESCRIPTION'	An externally visible description of the share.
PROP_SHARE_SPLIT_SIZE	VARCHAR2(10)	'SPLIT_SIZE'	The SPLIT_SIZE property determine how large the generated parquet files will be. The algorithm works as follows: 1. The SQL Optimizer estimates the size in bytes of a given segment (for example, a partition of a table or the results of a view). 2. The number of bytes is divided by the SPLIT_SIZE and rounded up to produce the "number of splits". 3. The data in the segment is then chunked into this "number of splits" using some method (for example ROWID, index columns, specified columns, etc.). 4. Each chunk of data is then converted into a parquet file. Note that the parquet file will generally be smaller than the SPLIT_SIZE due to the compression. In particular, SPLIT_SIZE determines the amount of data before compression.
PROP_SHARE_LOG_LEVEL	VARCHAR2(9)	'LOG_LEVEL'	The LOG_LEVEL property determines the amount of information that will be logged in the ALL_SHARE_EVENTS log.
PROP_SHARE_VERSION_ACCESS	VARCHAR2(14)	'VERSION_ACCESS'	The versions of a published share a recipient can see.

Share Recipient PAR Type

These constants are the valid values for the property `PROP_RECIPIENT_PAR_TYPE` in UPDATE_RECIPIENT_PROPERTY Procedure and GET_RECIPIENT_PROPERTY.

Name	Type	Value	Description
PAR_TYPE_FOLDER	PLS_INTEGER	1	PARs gives access to the entire folder.
PAR_TYPE_FILE	PLS_INTEGER	2	PARs gives access to one file at a time.

Share Recipient Properties

These constants are used in GET_SHARE_PROPERTY Function, UPDATE_DEFAULT_SHARE_PROPERTY Procedure, and UPDATE_SHARE_PROPERTY Procedure.

Name	Type	Value	Description
PROP_RECIPIENT_PAR_LIFETIME	VARCHAR2(12)	'PAR_LIFETIME'	The lifetime of Pre-authenticated Request URLs sent to the recipient through the delta sharing API. The default is three hours: '00 03:00:00'
PROP_RECIPIENT_MIN_PAR_LIFETIME	VARCHAR2(16)	'MIN_PAR_LIFETIME'	The minimum guaranteed lifetime of a Pre-authenticated Request URLs sent to the recipient through the delta sharing API. If there is an existing PAR for the same parquet file, it is reused only if the remaining life exceeds this minimum value. The default is 2 1/2 hours: '00 02:30:00'
PROP_RECIPIENT_MAX_PAR_LIFETIME	VARCHAR2(16)	'MAX_PAR_LIFETIME'	An administrator setting that defines the maximum allowed par lifetime for any recipient. The default is 1 day: '01 00:00:00'
PROP_RECIPIENT_TOKEN_LIFETIME	VARCHAR2(14)	'TOKEN_LIFETIME'	A string, in INTERVAL DAY TO SECOND format, representing the lifetime of a delta sharing bearer token. The default is one hour: '01 00:00:00'.
PROP_RECIPIENT_MAX_TOKEN_LIFETIME	VARCHAR2(18)	'MAX_TOKEN_LIFETIME'	An administrator setting that defines the maximum allowed bearer token lifetime for any recipient. The default is 90 days: '90 00:00:00'
PROP_RECIPIENT_EMAIL	VARCHAR2(5)	'EMAIL'	The email of the recipient. This is only required for delta sharing recipients.
PROP_RECIPIENT_DESCRIPTION	VARCHAR2(11)	'DESCRIPTION'	A textual description of the recipient.

Name	Type	Value	Description
PROP_RECIPIENT_LOG_LEVEL	VARCHAR2(9)	'LOG_LEVEL'	The logging level for the recipient. This controls what gets logged in USER_SHARE_RECIPIENT_EVENTS. The value should be one of the LOG_LEVEL_* constants.
PROP_RECIPIENT_SHARING_ID	VARCHAR2(10)	'SHARING_ID'	The sharing ID of the recipient. This is only required for Autonomous Database recipients.
PROP_RECIPIENT_PAR_TYPE	VARCHAR2(8)	'PAR_TYPE'	The type of pre-authenticated request to create. This should be one of PAR_TYPE_FOLDER (the default) or PAR_TYPE_FILE.
PROP_RECIPIENT_VERSION_ACCESS	VARCHAR2(14)	'VERSION_ACCESS'	Specifis what versions of published shares a recipient can see. One of VERSION_ACCESS_CURRENT or VERSION_ACCESS_ANY.

Share Table Properties

Constants used to indicate share table properties. These are used by GET_SHARE_TABLE_PROPERTY Function, and UPDATE_SHARE_TABLE_PROPERTY Procedure.

Name	Type	Value	Description
PROP_SHARE_TABLE_SPLIT_METHOD	VARCHAR2(12)	'SPLIT_METHOD'	Specifies how a segment should be split into different files.
SPLIT_METHOD_AUTO	VARCHAR2(4)	'AUTO'	Autonomous Database determines the way in which segments are split.
SPLIT_METHOD_RANGE	VARCHAR2(5)	'RANGE'	Split segments based on the value ranges. The exact ranges are determined by analyzing the data histograms. This method requires one or more split_columns to be specified.
SPLIT_METHOD_ROWID	VARCHAR2(5)	'ROWID'	Split segments based on ROWID.
SPLIT_METHOD_HASH	VARCHAR2(4)	'HASH'	Split segments based on HASH values. This method requires one or more split_columns to be specified.

Name	Type	Value	Description
SPLIT_METHOD_VALUE	VARCHAR2(5)	'VALUE'	Split segments based on distinct values. This method requires one or more split_columns to be specified.
PROP_SHARE_TABLE_SPLIT_COLUMNS	VARCHAR2(13)	'SPLIT_COLUMNS'	Specifies what columns are used for splitting. These are required for RANGE, VALUE, AND HASH methods. The value should be a comma delimited set of shared columns (for example, 'state,city' or '"STATE","CITY"').
PROP_SHARE_TABLE_ORDER_COLUMNS	VARCHAR2(13)	'ORDER_COLUMNS'	Specifies what columns are used for ordering when gathering parquet. The value should be a comma delimited set of shared columns (for example, 'state,city' or '"STATE","CITY"').
PROP_SHARE_TABLE_SHARE_COLUMNS	VARCHAR2(13)	'SHARE_COLUMNS'	Specifies what columns are shared. The value is a comma delimited set of shared columns. For example, 'state,city' or '"STATE","CITY"'.
PROP_SHARE_TABLE_SPLIT_SIZE	VARCHAR2(10)	'SPLIT_SIZE'	Override share level split_size for a specific table. The value is a number.
PROP_SHARE_TABLE_GATHER_STATS	VARCHAR2(12)	'GATHER_STATS'	If Autonomous Database will gather statistics, the value of 'GATHER_STATS' is 'YES'. Otherwise the value is 'NO'.

Name	Type	Value	Description
PROP_SHARE_TABLE_SPLIT_ROWS	VARCHAR2(10)	'SPLIT_ROWS'	The SPLIT_ROWS property is an alternative way to specify the amount of data to be stored in each parquet file. 1. Autonomous Database calculates the number of rows in a given segment (for example, a partition of a table or the results of a view). 2. The number of rows is divided by the SPLIT_ROWS and rounded up to produce the "number of splits". 3. The data in the segment is then chunked into this number of splits using some method (for example, ROWID, index columns, specified columns, etc.) 4. Each chunk of data is then converted into a parquet file. It is recommended that you use the SPLIT_SIZE property to control the number of splits since this reflects the amount of data in each row. A row with 900 columns of VARCHAR2(4000), for example, is much larger than a row with a single column of type NUMBER(1). Only use SPLIT_ROWS if you have accounted for the average amount of data in each row.

Name	Type	Value	Description
PROP_SHARE_TABLE_FLASHBACK	VARCHAR2(9)	'FLASHBACK'	The FLASHBACK property determines if the table will be published using flashback queries to ensure read consistency between the different export files. The value should be YES, NO, or AUTO. The AUTO setting will use flashback if it is available. The YES setting will cause the publication to fail if flashback is not possible on the table or view. The NO setting will not use flashback, which means that parquet files could represent different snapshots of the data.

Share Types

Constants used during share creation to identify the type of share.

See CREATE_SHARE Procedure.

Name	Type	Value	Description
SHARE_TYPE_VERSIONED	DBMS_ID	'VERSIONED'	Share is used for sharing versioned data.
SHARE_TYPE_LIVE	DBMS_ID	'LIVE'	Share is used for sharing live data.

Sharing ID Type

Used in ASSERT_SHARING_ID and GET_SHARING_ID.

Name	Type	Value	Description
SHARING_ID_TYPE_TENANCY	VARCHAR2(7)	'TENANCY'	The sharing ID represents an OCI Tenancy.
SHARING_ID_TYPE_COMPARTMENT	VARCHAR2(11)	'COMPARTMENT'	The sharing ID represents an OCI Compartment.
SHARING_ID_TYPE_DATABASE	VARCHAR2(8)	'DATABASE'	The sharing ID represents an OCI Autonomous Database.
SHARING_ID_TYPE_REGION	VARCHAR2(6)	'REGION'	The sharing ID represents an OCI Region.

Version Access

Possible values for the VERSION_ACCESS share property. This is used to implement delta "time travel". These are the valid values associated with PROP_SHARE_VERSION_ACCESS and PROP_RECIPIENT_VERSION_ACCESS.

These can be used to enable the "version" property in the delta sharing query endpoint, which allows recipients to select specific versions of the data instead of always receiving the CURRENT version. You need to enable this on both the share and the recipient.

For example:

```
UPDATE_SHARE_PROPERTY(
          share_name => '...',
          share_property => DBMS_SHARE.PROP_SHARE_VERSION_ACCESS,
          new_value => DBMS_SHARE.VERSION_ACCESS_ANY);

UPDATE_SHARE_PROPERTY(
          share_name => '...',
          share_property => DBMS_SHARE.PROP_RECIPIENT_VERSION_ACCESS,
          new_value => DBMS_SHARE.VERSION_ACCESS_ANY);
```

Name	Type	Value	Description
VERSION_ACCESS_CURRENT	PLS_INTEGER	1	A recipient can see only the CURRENT version. (Default)
VERSION_ACCESS_ANY	PLS_INTEGER	2	A recipient can choose any CURRENT or RETIRED version.

Autonomous Database for Experienced Oracle Database Users

This appendix provides information on using Autonomous Database for experienced Oracle Database users with Autonomous Database Serverless.

For equivalent information about using Oracle Database features and options with Autonomous Database on dedicated Exadata infrastructure, see Oracle Database Features in Dedicated Autonomous Database Deployments.

- About Autonomous Database for Experienced Oracle Database Users
  Autonomous Database configures and optimizes your database for you. You do not need to perform administration operations for configuring the database. SQL commands used for database administration such as `CREATE TABLESPACE` are not available. Similarly, other administrative interfaces and utilities such as `RMAN` are not available.
- Autonomous Database Views
  Autonomous Database provides several views that are not available in Oracle Database 19c. This topic lists the Autonomous Database specific views.
- Autonomous Database – Oracle Database Features
  Describes Oracle Database features available with Autonomous Database.
- Always Free Autonomous Database – Oracle Database 21c
  When you provision Always Free Autonomous Database you can select either Oracle Database 19c or Oracle Database 23ai.
- Always Free Autonomous Database – Oracle Database 23ai
  When you provision Always Free Autonomous Database you can select either Oracle Database 19c or Oracle Database 23ai.
- Autonomous Database RMAN Recovery Catalog
  You can use Oracle Autonomous Database as a Recovery Manager (RMAN) recovery catalog. A recovery catalog is a database schema that RMAN uses to store metadata about one or more Oracle databases.

- Notes for Users Migrating from Other Oracle Databases
  Describes information that is useful when you are migrating from other Oracle Databases to Oracle Autonomous Database.
- Database Features Unavailable in Autonomous Database
  Lists the Oracle Database features that are not available in Autonomous Database. Additionally, database features designed for administration are not available.
- Logical Partition Change Tracking and Materialized Views
  Describes information about the Logical Partition Change Tracking (LPCT) metadata framework and Query Rewrite with Logical Partition Change Tracking in Autonomous Database.

About Autonomous Database for Experienced Oracle Database Users

Autonomous Database configures and optimizes your database for you. You do not need to perform administration operations for configuring the database. SQL commands used for database administration such as `CREATE TABLESPACE` are not available. Similarly, other administrative interfaces and utilities such as `RMAN` are not available.

There are differences, depending on your workload: Data Warehouse, Transaction Processing, or JSON Database. See the following section appropriate to your workload:

- Data Warehouse Workload with Autonomous Database
- Transaction Processing and JSON Database Workloads with Autonomous Database
- Data Warehouse Workload with Autonomous Database
  Autonomous Database configures and optimizes your database for you, based on your workload.
- Transaction Processing and JSON Database Workloads with Autonomous Database
  Autonomous Database configures and optimizes your database for you, based on your workload.

Data Warehouse Workload with Autonomous Database

Autonomous Database configures and optimizes your database for you, based on your workload.

Characteristics of a database with Data Warehouse workload:

- The default data and temporary tablespaces for the database are configured automatically. Adding, removing, or modifying tablespaces is not allowed. Autonomous Database creates one tablespace or multiple tablespaces automatically depending on the storage size.
- The database character set is Unicode `AL32UTF8`. See Choose a Character Set for Autonomous Database for more information.
- Compression is enabled by default. Autonomous Database uses Hybrid Columnar Compression for all tables by default. You can specify different compression methods for your tables using the compression clause in your `CREATE TABLE` or `ALTER TABLE` commands.
- Oracle Database Result Cache is enabled by default for all SQL statements.

Accessing a database:

- You do not have direct access to the database node. You can create and drop directories with `CREATE DIRECTORY` and `DROP DIRECTORY`, as described in Creating and Managing Directories on Autonomous Database.

You can use `DBMS_CLOUD` procedures such as `DBMS_CLOUD.DELETE_FILE`, `DBMS_CLOUD.GET_OBJECT`, and `DBMS_CLOUD.PUT_OBJECT` with files and objects. You do not have direct access to the local file system.

Parallel Execution with Data Warehouse workload:

- Parallelism is determined by the database service. See Database Service Names for Autonomous Data Warehouse for details for parallelism support for each database service.
- When you want to disable parallel DML operations in your session, use the following SQL command:

```
ALTER SESSION DISABLE PARALLEL DML;
```

  See *VLDB and Partitioning Guide* for more information on parallel DML operations.
- Manage DML Performance and Compression for Data Warehouse Workloads
- Create Staging Tables for Data Warehouse Workloads
  Autonomous Database supports staging tables that are optimized for loading data into a data warehouse.

Manage DML Performance and Compression for Data Warehouse Workloads

Autonomous Database with Data Warehouse workloads uses Hybrid Columnar Compression for all tables by default. This gives the best compression ratio and optimal performance for direct-path load operations like the loads done using the `DBMS_CLOUD` package. If you perform DML operations like `UPDATE` and `MERGE` on your tables these may cause the compression ratio for the affected rows to decrease leading to larger table sizes. These operations may also perform slower compared to the same operations on an uncompressed table.

For the best compression ratio and optimal performance Oracle recommends using bulk operations like direct-path loads and `CREATE TABLE AS SELECT` statements. But, if your workload requires frequent DML operations like `UPDATE` and `MERGE` on large parts of a table, you can create those tables as uncompressed tables to achieve better DML performance. For example, the following statement creates the table `SALES` as an uncompressed table:

```
CREATE TABLE sales (
    prod_id             NUMBER          NOT NULL,
    cust_id             NUMBER          NOT NULL,
    time_id             DATE            NOT NULL,
    channel_id          NUMBER          NOT NULL,
    promo_id            NUMBER          NOT NULL,
    quantity_sold       NUMBER(10,2)    NOT NULL,
    amount_sold         NUMBER(10,2)    NOT NULL)
NOCOMPRESS;
```

At any point in time you can use the `ALTER TABLE MOVE` statement to compress these tables without impacting queries accessing them. For example, the following statement compresses the table `SALES` using Hybrid Columnar Compression.

```
ALTER TABLE sales MOVE COLUMN STORE COMPRESS FOR QUERY HIGH;
```

Create Staging Tables for Data Warehouse Workloads

Autonomous Database supports staging tables that are optimized for loading data into a data warehouse.

A staging table is a table with the `STAGING` property set. This applies the following characteristics:

- Any form of compression is explicitly turned off and disallowed on a staging table for any data load. The command `ALTER TABLE COMPRESS` is not allowed.
- Setting the `STAGING` property on an existing table does not impact the storage of existing data but does impact future data loads.
- Autonomous Database uses dynamic sampling for statistics for tables with the staging property set, and does not collect statistics on staging tables.
- Dropping staging tables immediately removes the table, bypassing the recycle bin. Setting the `recyclebin` initialization parameter to the value `ON` does not enable the recycle bin.

The characteristics of Autonomous Database partitioned staging tables includes the above, plus the following:

- Any form of compression is explicitly turned off and disallowed on all of the table's partitions and subpartitions.
- You cannot change the default attributes of the table to use compress with `ALTER TABLE MODIFY DEFAULT ATTRIBUTES`.
- You cannot perform partition maintenance operations that move data and compress the data. For example, the following are not allowed when you try to apply compression: `ALTER TABLE` with `MOVE PARTITION`, `MERGE PARTITIONS`, `SPLIT PARTITION`, or `SPLIT SUBPARTITION`.
- You cannot repartition a table with `ALTER TABLE MODIFY PARTITION` and specify any resulting partition to be compressed.

Define staging tables when you create a table or by altering an existing table as follows:

1. Create a table with the `STAGING` property.

   For example:

   ```
   CREATE TABLE staging_table (col1 number, col2 varchar2(100)) FOR STAGING;

   CREATE TABLE part_staging_table (col1 number, col2 varchar2(100))
       PARTITION BYRANGE (col1)
             (PARTITION p1 VALUESLESS THAN (100),
              PARTITION pmax VALUESLESS THAN (MAXVALUE)) FOR STAGING;
   ```

2. Change an existing table to set the `STAGING` property.

   For example:

   ```
   ALTER TABLE staging_table FOR STAGING;
   ```

3. You can verify the `STAGING` property for a table using one of the following views: `USER_TABLES`, `ALL_TABLES`, or `DBA_TABLES`.

   In these views, the `STAGING` column indicates the staging table property, a value `YES` indicates a `STAGING` table, a value `NO` is shown for all other tables.

You can alter a table to remove the `STAGING` property. For example:

```
ALTER TABLE staging_table NOT FOR STAGING;
```

Note the following for altering a table with `NOT FOR STAGING`:

- After you alter a table with `NOT FOR STAGING`, the compression attribute and existing data are not affected and are kept as uncompressed until you explicitly alter the table and specify compression. You can change the table compression and `ALTER TABLE COMPRESS` is allowed.
- Altering a table with `NOT FOR STAGING` does not trigger statistics collection. After you change the table property with `NOT FOR STAGING`, you can collect statistics, either manually or automatically.
- After you alter a table with `NOT FOR STAGING`, when the recycle bin is enabled dropping the table puts the table in the recycle bin.

Transaction Processing and JSON Database Workloads with Autonomous Database

Autonomous Database configures and optimizes your database for you, based on your workload.

Characteristics of Autonomous Database with Transaction Processing or JSON Database workloads:

- The default data and temporary tablespaces for the database are configured automatically. Adding, removing, or modifying tablespaces is not allowed. Autonomous Database creates one tablespace or multiple tablespaces automatically depending on the storage size.
- The database character set is Unicode `AL32UTF8`. See Choose a Character Set for Autonomous Database for more information.
- Compression is not enabled by default but Autonomous Database honors a compression clause if compression is specified on a table.

Accessing a database:

- You do not have direct access to the database node. You can create and drop directories with `CREATE DIRECTORY` and `DROP DIRECTORY`, as described in Create and Manage Directories.

  You can use `DBMS_CLOUD` procedures such as `DBMS_CLOUD.DELETE_FILE`, `DBMS_CLOUD.GET_OBJECT`, and `DBMS_CLOUD.PUT_OBJECT` with files and objects. You do not have direct access to the local file system.

Parallel Execution with Transaction Processing or JSON Database workloads:

- Parallelism is determined by the database service you use. See Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database for details for parallelism support for each database service.
- When you want to run DML operations in parallel and the database service you are using allows this, you can enable parallel DML in your session using the following SQL command:

  ```
  ALTER SESSION ENABLE PARALLEL DML;
  ```

  See *VLDB and Partitioning Guide* for more information on parallel DML operations.

- If you create an index manually and specify the `PARALLEL` clause, the `PARALLEL` attribute remains after the index is created. In this case SQL statements can run in parallel unbeknownst to the end user.

  To specify serial execution, change the `INDEX` parallel clause to `NOPARALLEL` or set the `PARALLEL` degree attribute to 1 to specify serial execution:

  ```
  ALTER INDEX index_name NOPARALLEL;
  ```

  or

  ```
  ALTER INDEX index_name PARALLEL 1;
  ```

Autonomous Database Views

Autonomous Database provides several views that are not available in Oracle Database 19c. This topic lists the Autonomous Database specific views.

- Track Table and Partition Scan Access with Autonomous Database Views
  Oracle Autonomous Database tracks the scan count for tables and partitions. Use the table access stats data dictionary and dynamic views to retrieve scan count information.
- Track Oracle Cloud Infrastructure Resources, Cost and Usage Reports with Autonomous Database Views
  Oracle Autonomous Database tracks the Oracle Cloud Infrastructure resources, cost and usage reports. You can access these reports using the OCI views.
- Pipeline Views
  Lists details for the `DBMS_CLOUD_PIPELINE` views.
- DBMS_CLOUD_AI Views
  The `DBMS_CLOUD_AI` package uses the following views.
- DBMS_CLOUD_FUNCTION Views
  The `DBMS_CLOUD_FUNCTION` package uses the following views.
- DBMS_CLOUD_LINK Views
  The `DBMS_CLOUD_LINK` package uses the following views.
- DBMS_DATA_ACCESS Views
  Describes the Autonomous Database dynamic performance views you can use with the `DBMS_DATA_ACCESS` package to monitor Pre-Authenticated Request (PAR) URL usage.
- Oracle Cloud Infrastructure Logging Interface Views
  The Oracle Cloud Infrastructure (OCI) logging interface views enable authorized users to access log data through a set of views.
- Real Application Testing Capture Replay Views
  Oracle Real Application Testing capture replay views.
- Stripe Views on Autonomous Database
  Stripe is an online payment processing and credit card processing platform for businesses. Users can query views created on top of Stripe APIs using the DBMS_CLOUD package to get Stripe information such as invoices, subscriptions, and customers.

Track Table and Partition Scan Access with Autonomous Database Views

Oracle Autonomous Database tracks the scan count for tables and partitions. Use the table access stats data dictionary and dynamic views to retrieve scan count information.

- GV$TABLE_ACCESS_STATS and V$TABLE_ACCESS_STATS Views
  The GV$TABLE_ACCESS_STATS and V$TABLE_ACCESS_STATS views list the scan count for tables and partitions. The scan data collection begins at instance startup time.
- ALL_TABLE_ACCESS_STATS and DBA_TABLE_ACCESS_STATS Views
  The ALL_TABLE_ACCESS_STATS and DBA_TABLE_ACCESS_STATS views list the scan count for tables and partitions. The scan data collection begins at instance startup time.
- USER_TABLE_ACCESS_STATS View
  The USER_TABLE_ACCESS_STATS view lists the scan count for the user's tables and partitions. The scan data collection begins at instance startup time.

GV$TABLE_ACCESS_STATS and V$TABLE_ACCESS_STATS Views

The GV$TABLE_ACCESS_STATS and V$TABLE_ACCESS_STATS views list the scan count for tables and partitions. The scan data collection begins at instance startup time.

Column	Datatype	Description
READ_COUNT	NUMBER	Aggregated scan count since instance startup
OBJECT_ID	NUMBER	Object ID of the table or partition
INST_ID	NUMBER	Instance number where table/partition was scanned This column (INST_ID) is only shown in GV$TABLE_ACCESS_STATS
CON_ID	NUMBER	Container ID of the database

ALL_TABLE_ACCESS_STATS and DBA_TABLE_ACCESS_STATS Views

The ALL_TABLE_ACCESS_STATS and DBA_TABLE_ACCESS_STATS views list the scan count for tables and partitions. The scan data collection begins at instance startup time.

> **Note:**
>
> The ALL_TABLE_ACCESS_STATS and DBA_TABLE_ACCESS_STATS views do not list scan count information for Oracle-maintained schemas.

Column	Datatype	Description
TABLE_OWNER	VARCAR2(128)	Owner of the table
TABLE_NAME	VARCAR2(128)	Name of the table
PARTITION_NAME	VARCAR2(128)	Name of the partition A NULL value specifies a non-partitioned table
INSTANCE_ID	NUMBER	Instance number where table or partition was scanned
READ_COUNT	NUMBER	Aggregated scan count since instance startup

USER_TABLE_ACCESS_STATS View

The USER_TABLE_ACCESS_STATS view lists the scan count for the user's tables and partitions. The scan data collection begins at instance startup time.

Column	Datatype	Description
TABLE_NAME	VARCAR2(128)	Name of the table
PARTITION_NAME	VARCAR2(128)	Name of the partition A NULL value specifies a non-partitioned table
INSTANCE_ID	NUMBER	Instance number where table/partition was scanned
READ_COUNT	NUMBER	Aggregated scan count since instance startup

Track Oracle Cloud Infrastructure Resources, Cost and Usage Reports with Autonomous Database Views

Oracle Autonomous Database tracks the Oracle Cloud Infrastructure resources, cost and usage reports. You can access these reports using the OCI views.

- Prerequisite Steps to Use OCI Resource Views
  Describes the prerequisite steps you must perform to use OCI resource views on Autonomous Database.
- OCI_AUTONOMOUS_DATABASES View
  OCI_AUTONOMOUS_DATABASES describes all the Oracle Cloud Infrastructure Autonomous Databases in the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.
- OCI_BUDGET_ALERT_RULES View
  OCI_BUDGET_ALERT_RULES describes all the Oracle Cloud Infrastructure budget alert rules in the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.
- OCI_BUDGET_SUMMARY View
  OCI_BUDGET_SUMMARY describes all the Oracle Cloud Infrastructure budget summaries in the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.
- OCI_COST_DATA View
  OCI_COST_DATA describes all the Oracle Cloud Infrastructure cost data for the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.
- OCI_OBJECTSTORAGE_BUCKETS View
  OCI_OBJECTSTORAGE_BUCKETS describes all the Oracle Cloud Infrastructure object storage buckets in the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.
- OCI_USAGE_DATA View
  OCI_USAGE_DATA describes all the Oracle Cloud Infrastructure usage data for the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.

Prerequisite Steps to Use OCI Resource Views

Describes the prerequisite steps you must perform to use OCI resource views on Autonomous Database.

> **Note:**
>
> Only ADMIN user has access to the OCI resource views by default. To access these views as another user, the ADMIN must grant READ privileges.

To query an OCI resource view, do the following:

1. Create a dynamic group that includes your Autonomous Database instance and define the required policies to access a view.

   For example, the Autonomous Database instance is specified in the `resource.id` parameter with an OCID:

   ```
   resource.id = '<your_Autonomous_Database_instance_OCID>'
   ```

   Each view shows the details for the policy that you must define to query the view.

   See Perform Prerequisites to Use Resource Principal with Autonomous Database for details on creating a dynamic group and defining policies.

   For example, to access all of the views, define the following policy:

   ```
   Define tenancy usage-report as
   ocid1.tenancy.oc1..aaaaaaaaned4fkpkisbwjlr56u7cj63lf3wffbilvqknstgtvzub7vhq
   kggq
   Endorse dynamic-group <group-name> to read objects in tenancy usage-report
   Allow dynamic-group <group-name> to read buckets in tenancy
   Allow dynamic-group <group-name> to read autonomous-database in tenancy
   Allow dynamic-group <group-name> to read usage-budgets in tenancy
   ```

   > **Note:**
   >
   > Do not replace the OCID in this policy with another OCID. This `usage-report` OCID provides the Oracle Cloud Infrastructure usage data for your tenancy.

2. Verify that resource principal is enabled for the ADMIN user on the Autonomous Database instance.

   ```
   SELECT owner, credential_name FROM dba_credentials
      WHERE credential_name = 'OCI$RESOURCE_PRINCIPAL' AND owner = 'ADMIN';

   OWNER CREDENTIAL_NAME
   ----- --------------------
   ADMIN OCI$RESOURCE_PRINCIPAL
   ```

If the resource principal is not enabled, then enable the resource principal:

```
EXEC DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL();
```

See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Run a query on an OCI resource view.

   For example:

```
SELECT NAME, APPROXIMATESIZE FROM OCI_OBJECTSTORAGE_BUCKETS;
SELECT * FROM OCI_USAGE_DATA;
```

OCI_AUTONOMOUS_DATABASES View

`OCI_AUTONOMOUS_DATABASES` describes all the Oracle Cloud Infrastructure Autonomous Databases in the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.

To query this view you need a dynamic group that includes your Autonomous Database instance and the following policy defined on that dynamic group:

```
Allow dynamic-group <group-name> to read autonomous-database in tenancy
```

This policy lets you list all Autonomous Databases in your tenancy. Optionally you can restrict it to list Autonomous Databases in a given compartment:

```
Allow dynamic-group <group-name> to read autonomous-database in compartment
<compartment-name>
```

Column	Datatype	Description
DISPLAYNAME	VARCHAR2	The user friendly name for the Autonomous Database
REGION	VARCHAR2	Region Name
COMPARTMENTID	VARCHAR2	The OCID of the compartment
ID	VARCHAR2	The OCID of the Autonomous Database
DBNAME	VARCHAR2	The database name
LIFECYCLESTATE	VARCHAR2	The current state of the Autonomous Database
TIMECREATED	VARCHAR2	The date and time the Autonomous Database was created
DATASTORAGESIZEINTBS	VARCHAR2	The quantity of data in the database in terabytes
LICENSEMODEL	VARCHAR2	The Oracle license model that applies to the Autonomous Database
SERVICECONSOLEURL	VARCHAR2	The URL of the Service Console for the Autonomous Database
APEXDETAILS	CLOB	Information about Oracle APEX Application Development
AREPRIMARYWHITELISTEDIPSUSED	VARCHAR2	Primary White Listed IPs
AUTONOMOUSCONTAINERDATABASEID	VARCHAR2	The Autonomous Container Database OCID
AUTONOMOUSMAINTENANCESCHEDULETYPE	VARCHAR2	Maintenance Schedule Type

Column	Datatype	Description
AVAILABLEUPGRADEVERSIONS	VARCHAR2	List of Oracle Database versions available for a database upgrade
BACKUPCONFIG	CLOB	Autonomous Database Backup Config
CONNECTIONSTRINGS	CLOB	Autonomous Database Connection Strings
CONNECTIONURLS	CLOB	Autonomous Database Connection URLs
CPUCORECOUNT	NUMBER	The number of OCPU cores to be made available to the database
CUSTOMERCONTACTS	CLOB	The Customer Contacts
DATASAFESTATUS	VARCHAR2	Status of the Data Safe registration for this Autonomous Database
DATASTORAGESIZEINGBS	NUMBER	The quantity of data in the database in gigabytes
DBVERSION	VARCHAR2	The Oracle Database version for the Autonomous Database
DATAGUARDREGIONTYPE	VARCHAR2	The Autonomous Data Guard region type of the Autonomous Database
DBWORKLOAD	VARCHAR2	The Autonomous Database workload type
DEFINEDTAGS	CLOB	Defined tags for the resource
FAILEDDATARECOVERYINSECONDS	NUMBER	Indicates the number of seconds of data loss for an Autonomous Data Guard failover
FREEFORMTAGS	CLOB	Free form tags for the resource
INFRASTRUCTURETYPE	VARCHAR2	The infrastructure type this resource belongs to
ISACCESSCONTROLENABLED	VARCHAR2	Indicates if the database level access control is enabled
ISAUTOSCALINGENABLED	VARCHAR2	Indicates if auto scaling is enabled for the Autonomous Database
ISDATAGUARDENABLED	VARCHAR2	Indicates whether the Autonomous Database has a local Autonomous Data Guard enabled
ISDEDICATED	VARCHAR2	True if the database uses dedicated Exadata infrastructure
ISFREETIER	VARCHAR2	Indicates if this is an Always Free resource
ISMTLSCONNECTIONREQUIRED	VARCHAR2	Indicates whether the Autonomous Database requires mTLS connections
ISPREVIEW	VARCHAR2	Indicates if the Autonomous Database version is a preview version
ISREFRESHABLECLONE	VARCHAR2	Indicates whether the Autonomous Database is a refreshable clone
KEYHISTORYENTRY	CLOB	Key History Entry
KEYSTOREID	VARCHAR2	The OCID of the key store
KEYSTOREWALLETNAME	VARCHAR2	The wallet name for Oracle Cloud Infrastructure Vault
KMSKEYID	VARCHAR2	The OCID of the key container that is used as the master encryption key
KMSKEYLIFECYCLEDETAILS	VARCHAR2	Customer managed key lifecycle details
LIFECYCLEDETAILS	VARCHAR2	Information about the current lifecycle state
NSGIDS	CLOB	A list of the OCIDs of the network security groups NSGs
OCPUCOUNT	NUMBER	The number of OCPU cores to be made available to the database
OPENMODE	VARCHAR2	The Autonomous Database open mode
OPERATIONSINSIGHTSSTATUS	VARCHAR2	Status of Operations Insights for this Autonomous Database

Column	Datatype	Description
PEERDBIDS	VARCHAR2	The list of OCIDs of standby databases located in Autonomous Data Guard
PERMISSIONLEVEL	CLOB	The Autonomous Database permission level
PRIVATEENDPOINT	VARCHAR2	The private endpoint for the resource
PRIVATEENDPOINTIP	VARCHAR2	The private endpoint IP address for the resource
PRIVATEENDPOINTLABEL	VARCHAR2	The private endpoint label for the resource
REFRESHABLEMODE	VARCHAR2	The refresh mode of the clone
REFRESHABLESTATUS	VARCHAR2	The refresh status of the clone
ROLE	VARCHAR2	The Autonomous Data Guard role
SOURCEID	VARCHAR2	The OCID of the source Autonomous Database that was cloned
SQLWEBDEVELOPERURL	VARCHAR2	The Database Actions (SQL Developer Web) URL for the Autonomous Database
STANDBYDB	CLOB	Autonomous Database Standby Summary
STANDBYWHITELISTEDIPS	CLOB	The client IP access control list
SUBNETID	VARCHAR2	The OCID of the subnet the resource is associated with
SUPPORTEDREGIONSTOCLONETO	CLOB	The list of regions that support the creation of Autonomous Data Guard
SYSTEMTAGS	CLOB	System tags for this resource
TIMEDATAGUARDROLECHANGED	VARCHAR2	The date and time the Autonomous Data Guard role was switched
TIMEDELETIONOFFREEAUTONOMOUSDATABASE	NUMBER	Time deletion of Free Autonomous Database
TIMELOCALDATAGUARDENABLED	VARCHAR2	The date and time that Autonomous Data Guard was enabled for the Autonomous Database
TIMEMAINTENANCEBEGIN	VARCHAR2	The date and time when maintenance will begin
TIMEMAINTENANCEEND	VARCHAR2	The date and time when maintenance will end
TIMEOFLASTFAILOVER	VARCHAR2	The timestamp of the last failover operation
TIMEOFLASTREFRESH	VARCHAR2	The date and time of the last refresh
TIMEOFLASTREFRESHPOINT	VARCHAR2	The refresh point timestamp
TIMEOFLASTSWITCHOVER	VARCHAR2	The timestamp of the last switchover operation for the Autonomous Database
TIMEOFNEXTREFRESH	VARCHAR2	The date and time of next refresh
TIMERECLAMATIONOFFREEAUTONOMOUSDATABASE	VARCHAR2	The date and time the Always Free database
USEDDATASTORAGESIZEINTBS	NUMBER	The amount of storage that has been used in terabytes
VAULTID	VARCHAR2	The OCID of the Oracle Cloud Infrastructure Vault
WHITELISTEDIPS	CLOB	The client IP access control list

OCI_BUDGET_ALERT_RULES View

`OCI_BUDGET_ALERT_RULES` describes all the Oracle Cloud Infrastructure budget alert rules in the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.

Queries against this view return results only if you have budgets and budget alerts created in your tenancy.

See Budgets Overview for more information.

To query this view you need a dynamic group that includes your Autonomous Database instance and the following policy defined on that dynamic group:

```
Allow dynamic-group <group-name> to read usage-budgets in tenancy
```

This policy lets you list budget summary and budget alerts in your tenancy (if you created a budget and a budget alert). Optionally you can restrict the result returned by querying the view to a given compartment:

```
Allow dynamic-group <group-name> to read usage-budgets in compartment <compartment-name>
```

Column	Datatype	Description
BUDGETID	VARCHAR2	The OCID of the budget
REGION	VARCHAR2	Region name
COMPARTMENTID	VARCHAR2	The compartment ID in which the bucket is authorized
DEFINEDTAGS	CLOB	Defined tags for the resource
DESCRIPTION	VARCHAR2	The description of the alert rule
DISPLAYNAME	VARCHAR2	The name of the alert rule
FREEFORMTAGS	CLOB	Free-form tags for the resource
ID	VARCHAR2	The OCID of the alert rule
LIFECYCLESTATE	VARCHAR2	The current state of the alert rule
MESSAGE	VARCHAR2	The custom message that will be sent when the alert is triggered
RECIPIENTS	VARCHAR2	The audience that receives the alert when it triggers
THRESHOLD	NUMBER	The threshold for triggering the alert
THRESHOLDTYPE	VARCHAR2	The type of threshold
TIMECREATED	VARCHAR2	The time when the budget was created
TIMEUPDATED	VARCHAR2	The time when the budget was updated
TYPE	VARCHAR2	ACTUAL or FORECAST types of alert triggers
VERSION	NUMBER	The version of the alert rule

OCI_BUDGET_SUMMARY View

OCI_BUDGET_SUMMARY describes all the Oracle Cloud Infrastructure budget summaries in the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.

Queries against this view return results only if you have budgets created in your tenancy.

See Budgets Overview for more information.

To query this view you need a dynamic group that includes your Autonomous Database instance and the following policy defined on that dynamic group:

```
Allow dynamic-group <group-name> to read usage-budgets in tenancy
```

This policy lets you list budget summary and budget alerts in your tenancy (if you created a budget and a budget alert). Optionally you can restrict the result returned by querying the view to a given compartment:

```
Allow dynamic-group <group-name> to read usage-budgets in compartment <compartment-name>
```

Column	Datatype	Description
REGION	VARCHAR2	Region name
COMPARTMENTID	VARCHAR2	The OCID of the compartment
AMOUNT	NUMBER	The amount of the budget, expressed in the currency of a rate card
DEFINEDTAGS	CLOB	Defined tags for the resource
FREEFORMTAGS	CLOB	Free-form tags for the resource
DISPLAYNAME	VARCHAR2	The display name of the budget
LIFECYCLESTATE	VARCHAR2	The current state of the budget
ACTUALSPEND	NUMBER	The actual spend in currency for the current budget cycle
ALERTRULECOUNT	NUMBER	The total number of alert rules in the budget
BUDGETPROCESSINGPERIODSTARTOFFSET	NUMBER	The number of days offset from the first day of the month, at which the budget processing period starts
DESCRIPTION	VARCHAR2	The description of the budget
FORECASTEDSPEND	NUMBER	The forecasted spend in currency by the end of the current budget cycle
ID	VARCHAR2	The OCID of the budget
RESETPERIOD	VARCHAR2	The reset period for the budget
TARGETS	CLOB	The list of targets on which the budget is applied
TARGETCOMPARTMENTID	VARCHAR2	Target compartment OCID
TARGETTYPE	VARCHAR2	The type of target on which the budget is applied
TIMECREATED	VARCHAR2	The time the budget was created
TIMESPENDCOMPUTED	VARCHAR2	The time the budget spend was last computed
TIMEUPDATED	VARCHAR2	The time the budget was updated
VERSION	VARCHAR2	The version of the budget

OCI_COST_DATA View

`OCI_COST_DATA` describes all the Oracle Cloud Infrastructure cost data for the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.

To query this view you need a dynamic group that includes your Autonomous Database instance and the following policy defined on that dynamic group:

```
Define tenancy usage-report as
ocid1.tenancy.oc1..aaaaaaaaned4fkpkisbwjlr56u7cj63lf3wffbilvqknstgtvzub7vhqkggq
Endorse dynamic-group <group-name> to read objects in tenancy usage-report
```

> **Note:**
>
> Do not replace the OCID in this policy with another OCID. This `usage-report` OCID provides the Oracle Cloud Infrastructure usage data for your tenancy.

Column	Datatype	Description
REFERENCE_NUMBER	VARCHAR2	Reference Number/Line identifier used for debugging and corrections

Column	Datatype	Description
TENANT_ID	VARCHAR2	The identifier (OCID) for the Oracle Cloud Infrastructure tenant
INTERVAL_USAGE_START	TIMESTAMP	The start time of the usage interval for the resource in UTC
INTERVAL_USAGE_END	TIMESTAMP	The end time of the usage interval for the resource in UTC
SERVICE_NAME	VARCHAR2	The service that the resource is in
COMPARTMENT_ID	VARCHAR2	The ID of the compartment that contains the resource
COMPARTMENT_NAME	VARCHAR2	The name of the compartment that contains the resource
REGION	VARCHAR2	The region that contains the resource
AVAILABILITY_DOMAIN	VARCHAR2	The availability domain that contains the resource
RESOURCE_ID	VARCHAR2	The identifier for the resource
BILLED_QUANTITY	VARCHAR2	The quantity of the resource that has been billed over the usage interval
BILLED_QUANTITY_OVERAGE	VARCHAR2	The usage quantity for which you were billed
SUBSCRIPTION_ID	VARCHAR2	A unique identifier associated with your commitment or subscription
PRODUCT_SKU	VARCHAR2	The Part Number for the resource in the line
PRODUCT_DESCRIPTION	VARCHAR2	The product description for the resource in the line
UNIT_PRICE	VARCHAR2	The cost billed to you for each unit of the resource used
UNIT_PRICE_OVERAGE	VARCHAR2	The cost per unit of usage for overage usage of a resource
MY_COST	VARCHAR2	The cost charged for this line of usage
MY_COST_OVERAGE	VARCHAR2	The cost billed for overage usage of a resource
CURRENCY_CODE	VARCHAR2	The currency code for your tenancy
BILLING_UNIT_READABLE	VARCHAR2	The unit measure associated with the usage/ billedQuantity in the line
SKU_UNIT_DESCRIPTION	VARCHAR2	The unit used for measuring billed quantity
OVERAGE_FLAG	CHAR	Flag used for overage usage
IS_CORRECTION	VARCHAR2	Used if the current line is a correction
BACK_REFERENCE_NUMBER	VARCHAR2	Data amendments and corrections reference
CREATED_BY	VARCHAR2	The user who created the service
CREATED_ON	TIMESTAMP	The time when the service was created
FREE_TIER_RETAINED	VARCHAR2	Is the service retained on free tier

OCI_OBJECTSTORAGE_BUCKETS View

OCI_OBJECTSTORAGE_BUCKETS describes all the Oracle Cloud Infrastructure object storage buckets in the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.

To query this view you need a dynamic group that includes your Autonomous Database instance and the following policy defined on that dynamic group:

```
Allow dynamic-group <group-name> to read buckets in tenancy
```

This policy lets you list object storage buckets in your tenancy. Optionally you can restrict the result returned by querying this view to a given compartment:

```
Allow dynamic-group <group-name> to read buckets in compartment <compartment-name>
```

Column	Datatype	Description
REGION	VARCHAR2	Region name
COMPARTMENTID	VARCHAR2	The compartment ID in which the bucket is authorized
NAMESPACE	VARCHAR2	The Object Storage namespace in which the bucket resides
APPROXIMATECOUNT	NUMBER	The approximate number of objects in the bucket
APPROXIMATESIZE	NUMBER	The approximate total size in bytes of all objects in the bucket
AUTOTIERING	VARCHAR2	The auto tiering status on the bucket
CREATEDBY	VARCHAR2	The OCID of the user who created the bucket
DEFINEDTAGS	CLOB	Defined tags for the resource
FREEFORMTAGS	CLOB	Free-form tags for the resource
ETAG	VARCHAR2	The entity tag (ETag) for the bucket
ID	VARCHAR2	The OCID of the bucket
ISREADONLY	VARCHAR2	Whether or not this bucket is read only
KMSKEYID	VARCHAR2	The OCID of a master encryption key
METADATA	VARCHAR2	Arbitrary string keys and values for user-defined metadata
NAME	VARCHAR2	The name of the bucket
OBJECTEVENTSENABLED	VARCHAR2	Whether or not events are emitted for object state changes in this bucket
OBJECTLIFECYCLEPOLICYETAG	VARCHAR2	The entity tag (ETag) for the live object lifecycle policy on the bucket
PUBLICACCESSTYPE	VARCHAR2	The type of public access enabled on this bucket
REPLICATIONENABLED	VARCHAR2	Whether or not this bucket is a replication source
STORAGETIER	VARCHAR2	The storage tier type assigned to the bucket
TIMECREATED	VARCHAR2	The date and time the bucket was created
VERSIONING	VARCHAR2	The versioning status on the bucket

OCI_USAGE_DATA View

OCI_USAGE_DATA describes all the Oracle Cloud Infrastructure usage data for the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.

To query this view you need a dynamic group that includes your Autonomous Database instance and the following policy defined on that dynamic group:

```
Define tenancy usage-report as
ocid1.tenancy.oc1..aaaaaaaaned4fkpkisbwjlr56u7cj63lf3wffbilvqknstgtvzub7vhqkggq
Endorse dynamic-group <group-name> to read objects in tenancy usage-report
```

> **Note:**
>
> Do not replace the OCID in this policy with another OCID. This `usage-report` OCID provides the Oracle Cloud Infrastructure cost and usage data for your tenancy.

Column	Datatype	Description
REFERENCE_NUMBER	VARCHAR2	Reference Number/Line identifier used for debugging and corrections
TENANT_ID	VARCHAR2	The identifier (OCID) for the Oracle Cloud Infrastructure tenant
INTERVAL_USAGE_START	TIMESTAMP	The start time of the usage interval for the resource in UTC
INTERVAL_USAGE_END	TIMESTAMP	The end time of the usage interval for the resource in UTC
SERVICE_NAME	VARCHAR2	The service that the resource is in
RESOURCE_NAME	VARCHAR2	The resource name used by the metering system
COMPARTMENT_ID	VARCHAR2	The ID of the compartment that contains the resource
COMPARTMENT_NAME	VARCHAR2	The name of the compartment that contains the resource
REGION	VARCHAR2	The region that contains the resource
AVAILABILITY_DOMAIN	VARCHAR2	The availability domain that contains the resource
RESOURCE_ID	VARCHAR2	The identifier for the resource
CONSUMED_QUANTITY	VARCHAR2	The quantity of the resource that has been consumed over the usage interval
BILLED_QUANTITY	VARCHAR2	The quantity of the resource that has been billed over the usage interval
CONSUMED_QUANTITY_UNITS	VARCHAR2	The unit for the consumed quantity and billed quantity
CONSUMED_QUANTITY_MEASURE	VARCHAR2	The measure for the consumed quantity and billed quantity
IS_CORRECTION	VARCHAR2	Used if the current line is a correction
BACK_REFERENCE_NUMBER	VARCHAR2	Data amendments and corrections reference
CREATED_BY	VARCHAR2	The user who created the service
CREATED_ON	TIMESTAMP	The time when the service was created
FREE_TIER_RETAINED	VARCHAR2	Is the service retained on free tier

Pipeline Views

Lists details for the `DBMS_CLOUD_PIPELINE` views.

- DBA_CLOUD_PIPELINES View
  `DBA_CLOUD_PIPELINES` displays the list of pipelines created in the database. The view contains one row for each pipeline.
- USER_CLOUD_PIPELINES View
  `USER_CLOUD_PIPELINES` displays the list of pipelines created in the database for the user. The view contains one row for each pipeline.
- DBA_CLOUD_PIPELINE_HISTORY View
  `DBA_CLOUD_PIPELINE_HISTORY` lists the pipeline scheduled jobs in the database. Use this view to monitor the health of the pipeline and detect failures in pipeline job runs.

- USER_CLOUD_PIPELINE_HISTORY View
  `USER_CLOUD_PIPELINE_HISTORY` lists the pipeline scheduled jobs in the database for the user. Use this view to monitor the health of the pipeline and detect failures in pipeline job runs.
- DBA_CLOUD_PIPELINE_ATTRIBUTES View
  `DBA_CLOUD_PIPELINE_ATTRIBUTES` lists the attributes of a pipelines in the database. Attributes control the configuration and behavior of the pipeline jobs.
- USER_CLOUD_PIPELINE_ATTRIBUTES View
  `USER_CLOUD_PIPELINE_ATTRIBUTES` lists the attributes of a pipelines in the database for the user. Attributes control the configuration and behavior of the pipeline.

DBA_CLOUD_PIPELINES View

`DBA_CLOUD_PIPELINES` displays the list of pipelines created in the database. The view contains one row for each pipeline.

Column	Datatype	NULL	Description
PIPELINE_ID	NUMBER	NOT NULL	Unique number assigned to the pipeline.
PIPELINE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline.
OWNER	VARCHAR2(128)	NOT NULL	Owner schema of the pipeline
PIPELINE_TYPE	VARCHAR2(7)	NOT NULL	Type of pipeline: `LOAD` or `EXPORT`
STATUS	VARCHAR2(7)	NOT NULL	Current status of pipeline: `STARTED` or `STOPPED`
DESCRIPTION	VARCHAR2(4000)		User specified description for the pipeline.
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Creation time for the pipeline.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time for the pipeline.
LAST_EXECUTION	TIMESTAMP(6) WITH TIME ZONE		Last successful execution time of pipeline job.
OPERATION_ID	NUMBER		Is the Operation ID for use with pipeline logs. See Monitor and Troubleshoot Pipelines for information on the `USER_LOAD_OPERATIONS` view for more information on data load operations.
STATUS_TABLE	VARCHAR2(128)		Status Table for pipeline progress. This is applicable for Load Pipelines.
ORACLE_MAINTAINED	CHAR(1)	NOT NULL	Indicates whether pipeline is Oracle maintained (`Y`) or not (`N`).

USER_CLOUD_PIPELINES View

`USER_CLOUD_PIPELINES` displays the list of pipelines created in the database for the user. The view contains one row for each pipeline.

Column	Datatype	NULL	Description
PIPELINE_ID	NUMBER	NOT NULL	Unique number assigned to the pipeline.
PIPELINE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline.
PIPELINE_TYPE	VARCHAR2(7)	NOT NULL	Type of pipeline: `LOAD` or `EXPORT`

Column	Datatype	NULL	Description
STATUS	VARCHAR2(7)	NOT NULL	Current status of pipeline: STARTED or STOPPED
DESCRIPTION	VARCHAR2(4000)		User specified description for the pipeline.
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Creation time for the pipeline.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time for the pipeline.
LAST_EXECUTION	TIMESTAMP(6) WITH TIME ZONE		Last successful execution time of pipeline job.
OPERATION_ID	NUMBER		Is the Operation ID for use with pipeline logs. See Monitor and Troubleshoot Pipelines for information on the USER_LOAD_OPERATIONS view for more information on data load operations.
STATUS_TABLE	VARCHAR2(128)		Status Table for pipeline progress. This is applicable for Load Pipelines.
ORACLE_MAINTAINED	CHAR(1)	NOT NULL	Indicates whether pipeline is Oracle maintained (Y) or not (N).

DBA_CLOUD_PIPELINE_HISTORY View

DBA_CLOUD_PIPELINE_HISTORY lists the pipeline scheduled jobs in the database. Use this view to monitor the health of the pipeline and detect failures in pipeline job runs.

Column	Datatype	NULL	Description
PIPELINE_ID	NUMBER	NOT NULL	Unique number assigned to the pipeline.
PIPELINE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline
OWNER	VARCHAR2(128)	NOT NULL	Owner schema of the pipeline.
PIPELINE_TYPE	VARCHAR2(7)		Type of pipeline: LOAD or EXPORT
JOB_ID	NUMBER	NOT NULL	Unique ID for the pipeline job execution.
JOB_NAME	VARCHAR2(1044)		Unique job name for the pipeline execution.
STATUS	VARCHAR2(30)		Status of the pipeline job.
START_DATE	TIMESTAMP(9) WITH TIME ZONE		Start time of the pipeline job execution.
END_DATE	TIMESTAMP(9) WITH TIME ZON		End time of the pipeline job execution.
INSTANCE_ID	NUMBER		Instance ID where the pipeline job ran.
SESSION_ID	VARCHAR2(128)		Session ID for the job session.
ERROR_NUMBER	NUMBER		Error number in the job (if encountered).
ERROR_MESSAGE	VARCHAR2(4000)		Error message in the job (if encountered).
CPU_USED	INTERVAL DAY(3) TO SECOND(2)		Number of CPU used by the pipeline job run.
JOB_OUTPUT	CLOB		Debug output of the pipeline job run.

USER_CLOUD_PIPELINE_HISTORY View

USER_CLOUD_PIPELINE_HISTORY lists the pipeline scheduled jobs in the database for the user. Use this view to monitor the health of the pipeline and detect failures in pipeline job runs.

Column	Data Type	NULL	Description
PIPELINE_ID	NUMBER	NOT NULL	Unique number assigned to the pipeline.
PIPELINE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline
PIPELINE_TYPE	VARCHAR2(7)		Type of pipeline: LOAD or EXPORT
JOB_ID	NUMBER	NOT NULL	Unique ID for the pipeline job execution.
JOB_NAME	VARCHAR2(1044)		Unique job name for the pipeline execution.
STATUS	VARCHAR2(30)		Status of the pipeline job.
START_DATE	TIMESTAMP(9) WITH TIME ZONE		Start time of the pipeline job execution.
END_DATE	TIMESTAMP(9) WITH TIME ZON		End time of the pipeline job execution.
INSTANCE_ID	NUMBER		Instance ID where the pipeline job ran.
SESSION_ID	VARCHAR2(128)		Session ID for the job session.
ERROR_NUMBER	NUMBER		Error number in the job (if encountered).
ERROR_MESSAGE	VARCHAR2(4000)		Error message in the job (if encountered).
CPU_USED	INTERVAL DAY(3) TO SECOND(2)		Number of CPU used by the pipeline job run.
JOB_OUTPUT	CLOB		Debug output of the pipeline job run.

DBA_CLOUD_PIPELINE_ATTRIBUTES View

DBA_CLOUD_PIPELINE_ATTRIBUTES lists the attributes of a pipelines in the database. Attributes control the configuration and behavior of the pipeline jobs.

Column	Datatype	NULL	Description
PIPELINE_ID	NUMBER	NOT NULL	Unique number assigned to the pipeline.
PIPELINE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline.
OWNER	VARCHAR2(128)	NOT NULL	Owner schema of the pipeline
PIPELINE_TYPE	VARCHAR2(7)	NOT NULL	Type of pipeline: LOAD or EXPORT
ATTRIBUTE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline attribute.
ATTRIBUTE_VALUE	CLOB		Value of the pipeline attribute.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time for the pipeline attribute.

USER_CLOUD_PIPELINE_ATTRIBUTES View

USER_CLOUD_PIPELINE_ATTRIBUTES lists the attributes of a pipelines in the database for the user. Attributes control the configuration and behavior of the pipeline.

Column	Datatype	NULL	Description
PIPELINE_ID	NUMBER	NOT NULL	Unique number assigned to the pipeline.

Column	Datatype	NULL	Description
PIPELINE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline.
PIPELINE_TYPE	VARCHAR2(7)	NOT NULL	Type of pipeline: LOAD or EXPORT
ATTRIBUTE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline attribute.
ATTRIBUTE_VALUE	CLOB		Value of the pipeline attribute.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time for the pipeline attribute.

DBMS_CLOUD_AI Views

The DBMS_CLOUD_AI package uses the following views.

- DBA_CLOUD_AI_PROFILES View
- USER_CLOUD_AI_PROFILES View
- DBA_CLOUD_AI_PROFILE_ATTRIBUTES View
- USER_CLOUD_AI_PROFILE_ATTRIBUTES View

DBA_CLOUD_AI_PROFILES View

The list of profiles created in the current database can be obtained by querying DBA_CLOUD_AI_PROFILES. The view displays the details of AI profiles in the database.

Only the ADMIN user is privileged to access DBA_CLOUD_AI_PROFILES. You cannot grant access to other users.

Column	Datatype	NULL	Description
PROFILE_ID	NUMBER	NOT NULL	Unique number assigned to the AI profile
PROFILE_NAME	VARCHAR2(128)	NOT NULL	Name of the AI profile
OWNER	VARCHAR2(128)	NOT NULL	Owner schema of the AI profile
STATUS	VARCHAR2(7)	NOT NULL	Current status of AI profile - ENABLED or DISABLED
DESCRIPTION	VARCHAR2(4000)	NOT NULL	User specified description for the AI profile
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Creation time for the AI profile.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time for the AI profile

USER_CLOUD_AI_PROFILES View

The list of profiles created in the current database can be obtained by querying USER_CLOUD_AI_PROFILES. The view displays the details of AI profiles in your schema.

Only the user is privileged to access USER_CLOUD_AI_PROFILES. You cannot grant access to other users.

Column	Datatype	NULL	Description
PROFILE_ID	NUMBER	NOT NULL	Unique number assigned to the AI profile
PROFILE_NAME	VARCHAR2(128)	NOT NULL	Name of the AI profile

Column	Datatype	NULL	Description
STATUS	VARCHAR2(7)	NOT NULL	Current status of AI profile - ENABLED or DISABLED
DESCRIPTION	VARCHAR2(4000)	-	User specified description for the AI profile
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Creation time for the AI profile
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time for the AI profile

DBA_CLOUD_AI_PROFILE_ATTRIBUTES View

Attributes of AI profile manage the configuration and behavior of the AI profile. The view displays details of all AI profiles in the database.

Only the ADMIN user is privileged to access DBA_CLOUD_AI_PROFILE_ATTRIBUTES. You cannot grant access to other users.

Column	Datatype	NULL	Description
PROFILE_ID	NUMBER	NOT NULL	Unique number assigned to the AI profile
PROFILE_NAME	VARCHAR2(128)	NOT NULL	Name of the AI profile
OWNER	VARCHAR2(128)	NOT NULL	Owner schema of the AI profile
ATTRIBUTE_NAME	VARCHAR2(128)	NOT NULL	Name of the AI profile attribute
ATTRIBUTE_VALUE	CLOB	-	Value of the AI profile attribute
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time of the AI profile attribute

USER_CLOUD_AI_PROFILE_ATTRIBUTES View

Attributes of AI profile manage the configuration and behavior of the AI profile. The view displays details of AI profiles in your schema.

Only the user is privileged to access USER_CLOUD_AI_PROFILE_ATTRIBUTES. You cannot grant access to other users.

Column	Datatype	NULL	Description
PROFILE_ID	NUMBER	NOT NULL	Unique number assigned to the AI profile
PROFILE_NAME	VARCHAR2(128)	NOT NULL	Name of the AI profile
ATTRIBUTE_NAME	VARCHAR2(128)	NOT NULL	Name of the AI profile attribute
ATTRIBUTE_VALUE	CLOB	-	Value of the AI profile attribute
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time of the AI profile attribute

DBMS_CLOUD_FUNCTION Views

The DBMS_CLOUD_FUNCTION package uses the following views.

- DBA_CLOUD_FUNCTION View
  The DBA_CLOUD_FUNCTION view lists all the functions created in your Autonomous Database.

- DBA_CLOUD_FUNCTION_CATALOG View
  The DBA_CLOUD_FUNCTION_CATALOG view lists all the catalogs created in an Autonomous Database instance.
- USER_CLOUD_FUNCTION View
  The USER_CLOUD_FUNCTION view lists all the functions created by the user.
- USER_CLOUD_FUNCTION_CATALOG View
  The USER_CLOUD_FUNCTION_CATALOG view lists all the catalogs created by the user.
- USER_CLOUD_FUNCTION_ERRORS View
  The USER_CLOUD_FUNCTION_ERRORS view lists the errors generated during the connection validation to the remote library location.

DBA_CLOUD_FUNCTION View

The DBA_CLOUD_FUNCTION view lists all the functions created in your Autonomous Database.

Column	Datatype	Description
OWNER	VARCHAR2(128)	Owner of the catalog
CATALOG_NAME	VARCHAR2(128)	Name of the catalog
APPLICATION_ID	VARCHAR2(2000)	OCI Function ID or NULL for AWS or Azure Application ID
CLOUD_FUNCTION_NAME	VARCHAR2(128)	Remote cloud function name
FUNCTION_NAME	VARCHAR2(128)	Function Name
FUNCTION_ID	VARCHAR2(2000)	OCI Function ID or AWS Function ARN or Azure Function ID
API_TYPE	VARCHAR2(128)	API type (Function or procedure)

DBA_CLOUD_FUNCTION_CATALOG View

The DBA_CLOUD_FUNCTION_CATALOG view lists all the catalogs created in an Autonomous Database instance.

Column	Datatype	Description
OWNER	VARCHAR2(128)	Owner of the catalog
CATALOG_NAME	VARCHAR2(128)	Name of the catalog
CLOUD_PROVIDER	VARCHAR2(128)	OCI Function ID or AWS Function ARN or Azure Application ID

USER_CLOUD_FUNCTION View

The USER_CLOUD_FUNCTION view lists all the functions created by the user.

Column	Datatype	Description
CATALOG_NAME	VARCHAR2(128)	Name of the catalog
APPLICATION_NAME	VARCHAR2(128)	OCI Application Name
APPLICATION_ID	VARCHAR2(2000)	OCI Application ID or Azure Application ID
CLOUD_FUNCTION_NAME	VARCHAR2(128)	Remote cloud function name

Column	Datatype	Description
FUNCTION_NAME	VARCHAR2(128)	Function Name
FUNCTION_ID	VARCHAR2(2000)	OCI Function ID or AWS Function ARN or Azure Function ID
API_TYPE	VARCHAR2(128)	API type (Function or procedure)

USER_CLOUD_FUNCTION_CATALOG View

The USER_CLOUD_FUNCTION_CATALOG view lists all the catalogs created by the user.

Column	Datatype	Description
CATALOG_NAME	VARCHAR2(128)	Name of the catalog
CLOUD_PROVIDER	VARCHAR2(128)	OCI Function ID or AWS Function ARN or Azure Application ID

USER_CLOUD_FUNCTION_ERRORS View

The USER_CLOUD_FUNCTION_ERRORS view lists the errors generated during the connection validation to the remote library location.

Column	Datatype	Description
CATALOG_NAME	VARCHAR2(128)	Name of the catalog
CREATE_TIMESTAMP	TIMESTAMP	Timestamp when error was generated
log_error_msg	CLOB	The error message

DBMS_CLOUD_LINK Views

The DBMS_CLOUD_LINK package uses the following views.

- DBA_CLOUD_LINK_ACCESS and ALL_CLOUD_LINK_ACCESS Views
- DBA_CLOUD_LINK_AUTHORIZATIONS View
- DBA_CLOUD_LINK_PRIVS and USER_CLOUD_LINK_PRIVS Views
- DBA_CLOUD_LINK_REGISTRATIONS and ALL_CLOUD_LINK_REGISTRATIONS Views
- V$CLOUD_LINK_ACCESS_STATS and GV$CLOUD_LINK_ACCESS_STATS Views

DBA_CLOUD_LINK_ACCESS and ALL_CLOUD_LINK_ACCESS Views

Displays details of the registered data sets an Autonomous Database instance is allowed to access.

Only the ADMIN user and users with the PDB_DBA role are privileged to access DBA_CLOUD_LINK_ACCESS. You cannot grant access to other users.

Column	Datatype	NULL	Description
NAMESPACE	VARCHAR2(128)	NOT NULL	Namespace of registered data set.
NAME	VARCHAR2(128)	NOT NULL	Name of registered data set.
CREATED	TIMESTAMP(6)	NOT NULL	Time/Date when the data set was created.

Column	Datatype	NULL	Description
LAST_MODIFIED	TIMESTAMP(6)	NOT NULL	Time when metadata of data set was last modified
AUTHORIZATION_REQUIRED	VARCHAR2(1)	NOT NULL	Data set requires additional authorization. Possible values are: "T" or "F".
DATA_SET_OWNER	VARCHAR2(128)	NULL	Data set owner.

DBA_CLOUD_LINK_AUTHORIZATIONS View

Displays details of which databases are authorized to access data sets. This applies to data sets that require additional authorization.

Only the ADMIN user and users with the PDB_DBA role are privileged to access DBA_CLOUD_LINK_AUTHORIZATIONS. You cannot grant access to other users.

Column	Datatype	NULL	Description
NAMESPACE	VARCHAR2(128)	NOT NULL	Namespace of registered data set.
NAME	VARCHAR2(128)	NOT NULL	Name of registered data set.
CLOUD_LINK_DATABASE_ID	VARCHAR2(40)	NOT NULL	Database ID of database that has been authorized for access.

DBA_CLOUD_LINK_PRIVS and USER_CLOUD_LINK_PRIVS Views

Displays details of the Cloud Link specific privileges, REGISTER, READ, AUTHORIZE, granted to specific users or to the current user.
USER_CLOUD_LINK_PRIVS columns are the same as those in DBA_CLOUD_LINK_PRIVS. (except for GRANTEE).

Only the ADMIN user and users with the PDB_DBA role are privileged to read from DBA_CLOUD_LINK_PRIVS. You cannot grant access to other users.

The privilege to read from USER_CLOUD_LINK_PRIVS view is granted to the ADMIN user, to the PDB_DBA role, and to the role DWROLE.

Column	Datatype	NULL	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of user to whom privilege is granted. This column is not available in USER_CLOUD_LINK_PRIVS.
PRIVILEGE	VARCHAR2(9)	NOT NULL	Privilege, one of: REGISTER, READ, AUTHORIZE, granted to user.
SCOPE	VARCHAR2(14)	NOT NULL	Scope of the privilege.

DBA_CLOUD_LINK_REGISTRATIONS and ALL_CLOUD_LINK_REGISTRATIONS Views

Displays details of all the data set registrations on an Autonomous Database instance.

Only the ADMIN user and users with the PDB_DBA role are privileged to access DBA_CLOUD_LINK_REGISTRATIONS. You cannot grant access to other users.

Column	Datatype	NULL	Description
NAMESPACE	VARCHAR2(128)	NOT NULL	Namespace of registered data set.
NAME	VARCHAR2(128)	NOT NULL	Name of registered data set.
ID	RAW(16)	NOT NULL	Unique ID of registered data set.
OWNER	VARCHAR2(128)	NOT NULL	Owner of object (table or view owner).
OBJECT_NAME	VARCHAR2(128)	NOT NULL	Name of object.
SCOPE	CLOB	NOT NULL	JSON document with details of who is allowed to access the registered data set.
DESCRIPTION	CLOB	NOT NULL	Description of registered data set.
CREATED	TIMESTAMP(6)	NOT NULL	Time/Date when the data set was created.
LAST_MODIFIED	TIMESTAMP(6)	NOT NULL	Time when metadata of data set was last modified
AUTHORIZATION_REQUIRED	VARCHAR2(1)	NOT NULL	Data set requires additional authorization. Possible values are: "T" or "F".
DATA_SET_OWNER	VARCHAR2(128)	NULL	Data set owner.

V$CLOUD_LINK_ACCESS_STATS and GV$CLOUD_LINK_ACCESS_STATS Views

Displays access to each registered data set owned by an Autonomous Database instance.

V$CLOUD_LINK_ACCESS_STATS tracks the elapsed time, DB CPU time, the number of rows retrieved, the number of executions, and the unique database identifier of the database accessing a registered data set.

In addition to the V$CLOUD_LINK_ACCESS_STATS information, the GV$CLOUD_LINK_ACCESS_STATS view contains an extra column named INST_ID of data type NUMBER. The INST_ID column displays the instance from which the associated V$ view information was obtained. The INST_ID column can be used as a filter to retrieve V$CLOUD_LINK_ACCESS_STATS information from a subset of available instances.

The unique database identifier does not disclose the OCIDs (Tenancy, Compartment, Database). This information can be used for auditing an Autonomous Database.

Column	Datatype	NULL	Description
INST_ID	NUMBER	NOT NULL	Instance ID
OWNER	VARCHAR2(128)	NOT NULL	Owner of object(table or view owner)
OBJECT_NAME	VARCHAR2(128)	NOT NULL	Name of object
DATA_SET_NAMESPACE	VARCHAR2(128)	NOT NULL	Namespace of registered data set
DATA_SET_NAME	VARCHAR2(128)	NOT NULL	Name of registered data set
SQL_ID	VARCHAR2(13)	NOT NULL	SQL ID of local SQL accessing a registered data set
LAST_ACTIVE_TIME	DATE	NOT NULL	Time when local SQL was last active
CLOUD_LINK_DATABASE_ID	VARCHAR2(512)	NOT NULL	Database identifier of database accessing a registered data set
ELAPSED_TIME	NUMBER	NOT NULL	Elapsed time (in microseconds) of local SQL
CPU_TIME	NUMBER	NOT NULL	CPU time (in microseconds) of local SQL

Column	Datatype	NULL	Description
USER_IO_WAIT_TIME	NUMBER	NOT NULL	User I/O Wait time (in microseconds) of local SQL
EXECUTIONS	NUMBER	NOT NULL	Number of executions of local SQL
FETCHES	NUMBER	NOT NULL	Number of fetches of local SQL
ROWS_PROCESSED	NUMBER	NOT NULL	Number of rows retrieved by local SQL
CON_ID	NUMBER	NOT NULL	Container ID

DBMS_DATA_ACCESS Views

Describes the Autonomous Database dynamic performance views you can use with the DBMS_DATA_ACCESS package to monitor Pre-Authenticated Request (PAR) URL usage.

- V$DATA_ACCESS_URL_STATS and GV$DATA_ACCESS_URL_STATS Views

V$DATA_ACCESS_URL_STATS and GV$DATA_ACCESS_URL_STATS Views

Displays access to each PAR URL owned by an Autonomous Database instance.

When a PAR URL is accessed, CPU time, User I/O wait time, are updated in the V$DATA_ACCESS_URL_STATS view.

The GV$CLOUD_LINK_ACCESS_STATS view contains the additional column named INST_ID of data type NUMBER. The INST_ID column displays the instance from which the associated V$ view information was obtained. The INST_ID column can be used as a filter to retrieve V$CLOUD_LINK_ACCESS_STATS information from a subset of available instances.

Column	Datatype	NULL	Description
INST_ID	NUMBER	NOT NULL	Instance ID
PRE_AUTH_URL_ID	VARCHAR2(128)	NOT NULL	Pre-Authenticated Request URL identifier
ELAPSED_TIME	NUMBER	NOT NULL	Elapsed time (in microseconds) of local SQL
CPU_TIME	NUMBER	NOT NULL	CPU time (in microseconds) of local SQL
USER_IO_WAIT_TIME	NUMBER	NOT NULL	User I/O Wait time (in microseconds) of local SQL
EXECUTIONS	NUMBER	NOT NULL	Number of executions of local SQL
FETCHES	NUMBER	NOT NULL	Number of fetches of local SQL
ROWS_PROCESSED	NUMBER	NOT NULL	Number of rows retrieved by local SQL
CON_ID	NUMBER	NOT NULL	Container ID

Oracle Cloud Infrastructure Logging Interface Views

The Oracle Cloud Infrastructure (OCI) logging interface views enable authorized users to access log data through a set of views.

Oracle Cloud Infrastructure (OCI) logging is the central logging solution for OCI services, including VCN flow logs, load balancer logs, object storage logs, and others within the added capability to include custom logs. The logs are not stored in Autonomous Database. Instead, the logs are dynamically retrieved based on various predicates provided by the user, including date range, log group, log name, and others.

- Oracle Cloud Infrastructure Logging Interface Overview
  Providing access to the log data inside an Autonomous Database in relational format provides a useful access method, along with the ability to get the logging data in relational or JSON format and enrich the analytics by joining with other data they may have in the Autonomous Database.
- OCI_LOG_LIST View
  Displays a list of all logs in the tenant. The view does not show log data, but provides the metadata about the logs in a tenancy. This metadata is used to properly specify parameter values for other logging view. If there is no logs under a log group, that log group will not be included in this view.
- OCI_LOG_DATA View
  The common view that provides access to all log types supported by the OCI Logging Service. The log content is in the `DATA` column.
- OCI_VCN_FLOWLOGS View
  View that provides details about traffic that passes through the user's VCN. This log enables users to audit traffic and troubleshoot security lists. Each flow log record reflects logged traffic in one direction of a connection between two endpoints. For example, a single TCP connection, you may have two records in the capture window: one for ingress traffic, and one for egress traffic.
- OCI_LBLOG_ACCESS View
  View that provides load balancer access logs capturing detailed information about requests sent to a load balancer. Each entry contains the time the request was received, client, intermediate HTTP proxy IP addresses, and times used by at the load balancer and backend to process the request.
- OCI_LBLOG_ERRORS View
  View that provides load balancer error logs capturing detailed information about requests related to troubleshooting and monitoring. Each entry contains information such as the time the request was received, error type, and additional details of the specific error.

Oracle Cloud Infrastructure Logging Interface Overview

Providing access to the log data inside an Autonomous Database in relational format provides a useful access method, along with the ability to get the logging data in relational or JSON format and enrich the analytics by joining with other data they may have in the Autonomous Database.

Oracle logging service provides access to logs from Oracle Cloud Infrastructure (OCI) resources. These logs include critical diagnostic information that describes how resources are performing and being accessed. There are varying log types for each Oracle service. To learn more about the logging service and supported OCI services, see Oracle Cloud Infrastructure Logging Overview.

Through the Oracle Cloud Infrastructure Logging Interface, log data can now be accessed through an Autonomous Database in relational format. Users can query different log data in

OCI across all compartments and regions. The implemented OCI views use the `OCI$RESOURCE_PRINCIPAL` credential.

To use these views, the following steps need to be done.

1. An Administrator enables use of the `OCI$RESOURCE_PRINCIPAL` credential in the database.
2. Users create a new dynamic group for the `OCI$RESOURCE_PRINCIPAL` credential.

To learn more about enabling the credential, setting up a dynamic group, and creating policy statements, see Use Resource Principal to Access Oracle Cloud Infrastructure Resources.

As an Administrator, enable the resource principal.

```
EXEC DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL();
```

The dynamic group must have the following rules that include the resource ID of the instance.

```
resource.id = '<resource.id>'
```

There are two policy statements required to access logging views.

```
-- For OCI Logging Views
Allow dynamic-group <group-name> to use logging-family in tenancy
Allow dynamic-group <group-name> to use compartments in tenancy
```

All the views have mandatory and optional predicate values (column names).

Predicate values are as follows:

> **Note:**
>
> `OCI_LOG_LIST` view does not have mandatory columns. If the `REGION` value is not provided, the view returns data for the home region.

- `REGION` (Mandatory)
- `COMPARTMENT_ID` (Mandatory)
- `LOG_GROUP_ID` (Mandatory)
- `LOG_ID` (Optional) - If the value is provided, query results will use the `LOG_ID`value. If no value provided, the value will be returned from the server.
- `START_TIME` (Optional) - If no value provided, return values are for past 5 minutes in GMT.
- `END_TIME` (Optional) - If no value provided, return values are for past 5 minutes in GMT.
- `SEARCH_CRITERIA` (Optional) - If no value provided, null will be returned.

Since there can be different log types within a single log group, it's recommended to specify `LOG_ID` in a predicate when using `OCI_VCN_FLOWLOGS`, `OCI_LBLOG_ACCESS`, and `OCI_LBLOG_ERRORS` views.

Notes and Restrictions:

- The common view `OCI_LOG_DATA` supports all log types, so `COMPARTMENT_ID`, `LOG_GROUP_ID`, and `REGION` are sufficient. Likewise, if a user uses the `LOG_ID` of a load

balancer with the OCI_VCN_FLOWLOGS view, it will either not return correct data or any data at all.

- For the OCI_LOG_LIST view, the select * from OCI_LOG_LIST will shows data only for the home region. If you need data for another region, you will have to use a REGION equality predicate in the WHERE clause.
- If the OCI_LOG_DATA, OCI_VCN_FLOWLOGS, OCI_LBLOG_ACCESS, and OCI_LBLOG_ERRORS, the select * from <view> gives an ORA-20000: compartment_id,log_group_id and region are mandatory predicates, please provide valid values for them as equality predicate in the WHERE clause. provide the required predicate values.
- The framework supports querying up to past 7 days (from current time) log data to query from the supported views. Any values for the predicates START_TIME and END_TIME predicates outside this range will result in ORA-20000: start_time and end_time should be from current_timestamp to current_timestamp-7.
- If you do not provide the listed mandatory predicates, you will see an ORA-20000: compartment_id,log_group_id and region are mandatory predicates, please provide valid values for them as equality predicate in the WHERE clause. Other columns can be used as predicates of any type (=, IN). The mandatory predicates can be only used once in a query, and they cannot be used multiple times with AND or NOT.
- Values for START_TIME and END_TIME have to be given in DD-MM-YY HH24:MI:SS format. If not provided in this format, the query will not return any data.

OCI_LOG_LIST View

Displays a list of all logs in the tenant. The view does not show log data, but provides the metadata about the logs in a tenancy. This metadata is used to properly specify parameter values for other logging view. If there is no logs under a log group, that log group will not be included in this view.

Column	Datatype	Description
REGION	VARCHAR2	OCI region name
COMPARTMENT_ID	VARCHAR2	Compartment OCID
LOG_GROUP_ID	VARCHAR2	LOG_GROUP OCID
LOG_GROUP_NAME	VARCHAR2	Log group name.
LOG_ID	VARCHAR2	LOG OCID
LOG_NAME	VARCHAR2	Log name
LOG_SERVICE	VARCHAR2	Log service

OCI_LOG_DATA View

The common view that provides access to all log types supported by the OCI Logging Service. The log content is in the DATA column.

Column	Datatype	Description
DATE_TIME	TIMESTAMP	Same as the oracle.ingestedtime field
ID	VARCHAR2	A random UUID for each log entry
SOURCE	VARCHAR2	Resource name that generated the log message

Column	Datatype	Description
SPEC_VERSION	NUMBER	Version of the CloudEvents specification this log message uses
LOG_TIME	TIMESTAMP	Log message generation time
TYPE	VARCHAR2	Log message type
COMPARTMENT_ID	VARCHAR2	Compartment OCID
INGESTED_TIME	TIMESTAMP	Log message ingest time
LOG_GROUP_ID	VARCHAR2	Log group OCID
LOG_ID	VARCHAR2	Log OCID
TENANT_ID	VARCHAR2	Tenant OCID of log object owner
DATA	CLOB	Log message
ORACLE_DETAILS	CLOB	Oracle-specific metadata

OCI_VCN_FLOWLOGS View

View that provides details about traffic that passes through the user's VCN. This log enables users to audit traffic and troubleshoot security lists. Each flow log record reflects logged traffic in one direction of a connection between two endpoints. For example, a single TCP connection, you may have two records in the capture window: one for ingress traffic, and one for egress traffic.

Column	Datatype	Description
REGION	VARCHAR2	OCI region name
DATE_TIME	TIMESTAMP	Same as the oracle.ingestedtime field
ID	VARCHAR2	A random UUID unique to each log entry
ACTION	VARCHAR2	Possible values of **ACCEPT** or **REJECT**
FLOW_ID	VARCHAR2	Hash of key fields (source and destination addresses, ports, and protocol)
STATUS	VARCHAR2	Possible values of **OK**, **NODATA**, or **SKIPDATA**
END_TIME	TIMESTAMP	Capture window end time
PACKETS	NUMBER	Number of packets recorded in capture window
VERSION	VARCHAR2	Flow log record schema number
BYTES_OUT	NUMBER	Number of bytes recorded in capture window
PROTOCOL	NUMBER	IANA protocol number
START_TIME	TIMESTAMP	Capture window start time
SOURCE_PORT	NUMBER	Source IANA port number
PROTOCOL_NAME	VARCHAR2	IANA protocol name
SOURCE_ADDRESS	VARCHAR2	IP address of the source in IPv4 or IPv6 notation
DESTINATION_PORT	NUMBER	Destination IANA port number
DESTINATION_ADDRESS	VARCHAR2	IP address of the destination in IPv4 or IPv6 notation

Column	Datatype	Description
TIME	TIMESTAMP	Same as START_TIME
TYPE	VARCHAR2	Log category
LOG_ID	VARCHAR2	Log OCID
TENANT_ID	VARCHAR2	Tenant OCID
VNIC_OCID	VARCHAR2	VNIC OCID
LOG_GROUP_ID	VARCHAR2	Log group OCID
INGESTED_TIME	TIMESTAMP	Time log ingested by OCI Logging
COMPARTMENT_ID	VARCHAR2	Log group compartment OCID
VNIC_SUBNET_OCID	VARCHAR2	VNIC subnet OCID
VNIC_COMPARTMENT_ID	VARCHAR2	VNIC compartment OCID
SOURCE	VARCHAR2	Resource name that generated log message
SPEC_VERSION	NUMBER	Version of the CloudEvents specification this log message uses

OCI_LBLOG_ACCESS View

View that provides load balancer access logs capturing detailed information about requests sent to a load balancer. Each entry contains the time the request was received, client, intermediate HTTP proxy IP addresses, and times used by at the load balancer and backend to process the request.

Column	Datatype	Description
REGION	VARCHAR2	OCI region name
DATE_TIME	TIMESTAMP	Same as the oracle.ingestedtime field
ID	VARCHAR2	A random UUID unique to each log entry
BACKEND_ADDRESS	VARCHAR2	IP address and port number of the backend server which processed the client request
BACKEND_CONNECT_TIME	NUMBER	Time spent (in seconds, with millisecond precision) to establish backend server connection
BACKEND_PROCESSING_TIME	NUMBER	Total time taken (in seconds, with millisecond precision) from the load balancer establishing a connection to a backend until it completes
BACKEND_STATUS_CODE	NUMBER	Status code of the response from the target
CLIENT_ADDRESS	VARCHAR2	IP address and port number of the requesting client
FORWARDED_FOR_ADDRESS	VARCHAR2	IP address of the client and http proxies between client and load balancer
HOST	VARCHAR2	Domain name which resolves to IP address assigned to the load balancer
LB_STATUS_CODE	NUMBER	Load balancer status code
LISTENER_NAME	VARCHAR2	Listener which received the incoming traffic request on the load balancer's IP address

Column	Datatype	Description
RECEIVED_BYTES	NUMBER	Total size of the request (in bytes) received from the client
REQUEST	VARCHAR2	Request line received from the client
REQUEST_PROCESSING_TIME	NUMBER	Total time taken (in seconds, with millisecond precision) from the load balancer receiving the request from the client until load balancer completes sending response to the client.
ROUTING_RULES_ENGINE_ERRORS	NUMBER	Routing rule engine error during policy evaluation of the request with a 0 (no error) or 1 (error). If an error occurred, requests are forwarded to the default backend attached to the listener.
ROUTING_RULES_MATCHED_RULE	VARCHAR2	Routing policy rule name, which was matched for this specific request
ROUTING_RULES_RULE_HITS	NUMBER	Number of routing rules evaluated to true for the request
ROUTING_RULES_RULE_MISSES	NUMBER	Number of routing rules evaluated to false for the request
SENT_BYTES	NUMBER	Total size of the request (in bytes) sent to the client
SSL_CIPHER	VARCHAR2	Negotiated SSL cipher between the client and the load balancer
SSL_PROTOCOL	VARCHAR2	Negotiated SSL protocol between the client and the load balancer
TIMESTAMP	TIMESTAMP	Log entry generation time
USER_AGENT	VARCHAR2	User agent used to send the request to the load balancer
TIME	TIMESTAMP	Log entry generation time
TYPE	VARCHAR2	Log category
LOG_ID	VARCHAR2	Log OCID
TENANT_ID	VARCHAR2	Tenant OCID
LOG_GROUP_ID	VARCHAR2	Log group OCID
RESOURCE_ID	VARCHAR2	Resource OCID
INGESTED_TIME	TIMESTAMP	Time log ingested by OCI Logging
COMPARTMENT_ID	VARCHAR2	Log group compartment OCID
SOURCE	VARCHAR2	Resource name that generated log message
SUBJECT	VARCHAR2	Subject of the log
SPEC_VERSION	NUMBER	Version of the `CloudEvents` specification this log message uses

OCI_LBLOG_ERRORS View

View that provides load balancer error logs capturing detailed information about requests related to troubleshooting and monitoring. Each entry contains information such as the time the request was received, error type, and additional details of the specific error.

Column	Datatype	Description
REGION	VARCHAR2	OCI region name
DATE_TIME	TIMESTAMP	Same as the oracle.ingestedtime field
ID	VARCHAR2	A random UUID unique to each log entry
ERROR_LOG_TYPE	VARCHAR2	Log type
ERROR_LOG_ERROR_DETAILS	VARCHAR2	Detailed description of the error message
DATA_TIMESTAMP	TIMESTAMP	Log entry generation time
TIME	TIMESTAMP	Log entry generation time
TYPE	VARCHAR2	Log category
LOG_ID	VARCHAR2	Log OCID
TENANT_ID	VARCHAR2	Tenant OCID
LOG_GROUP_ID	VARCHAR2	Log group OCID
RESOURCE_ID	VARCHAR2	Resource OCID
INGESTED_TIME	TIMESTAMP	Time log ingested by OCI Logging
COMPARTMENT_ID	VARCHAR2	Log group compartment OCID
SOURCE	VARCHAR2	Resource name that generated log message
SUBJECT	VARCHAR2	Subject of the log
SPEC_VERSION	NUMBER	Version of the CloudEvents specification this log message uses

Real Application Testing Capture Replay Views

Oracle Real Application Testing capture replay views.

- DBA_CAPTURE_REPLAY_HISTORY View
  The DBA_CAPTURE_REPLAY_HISTORY view provides historical information for all your workload captures and replays.
- DBA_CAPTURE_REPLAY_STATUS View
  The DBA_CAPTURE_REPLAY_STATUS view provides the most recent status for your workload. The values in the view are updated when you execute FINISH_WORKLOAD_CAPTURE, CANCEL_WORKLOAD_CAPTURE or REPLAY_WORKLOAD.

DBA_CAPTURE_REPLAY_HISTORY View

The DBA_CAPTURE_REPLAY_HISTORY view provides historical information for all your workload captures and replays.

Column	Datatype	Description
NAME	VARCHAR2(32)	Name of the workload capture or replay
UNIQUE_NAME	VARCHAR2(64)	Internal unique name for capture or replay
STATUS	VARCHAR2(8)	Workload capture or replay status
DATABASE_NAME	VARCHAR2(128)	Name of the database
DATABASE_GUID	RAW(16)	Unique id of the database
TYPE	VARCHAR2(10)	Type (capture or replay)

Column	Datatype	Description
START_TIME	DATE	Start time of the workload capture or replay.
END_TIME	DATE	End time of the workload capture or replay.
REPORT_URL	VARCHAR2(1000)	PAR URL path for capture or replay report.

DBA_CAPTURE_REPLAY_STATUS View

The DBA_CAPTURE_REPLAY_STATUS view provides the most recent status for your workload. The values in the view are updated when you execute FINISH_WORKLOAD_CAPTURE, CANCEL_WORKLOAD_CAPTURE or REPLAY_WORKLOAD.

Column	Datatype	Description
STATE	VARCHAR2(32)	Provides the current status for a workload capture or replay. For a workload capture, status value could be one of these: • CAPTURE *capture_name* STARTED • FINISHING CAPTURE • UPLOADING CAPTURE FILES • CANCELING CAPTURE • CANCEL CAPTURE *capture_name* COMPLETED • FINISH CAPTURE *capture_name* COMPLETED For a workload replay, status value could be one of these: • DOWNLOADING CAPTURE FILES • PROCESSING CAPTURE • INITIALIZING REPLAY • PREPARING REPLAY • STARTING REPLAY • UPLOADING REPLAY FILES • REPLAY *replay_name* COMPLETED
PROGRESS	VARCHAR2(8)	Percentage completed for download or upload.

Stripe Views on Autonomous Database

Stripe is an online payment processing and credit card processing platform for businesses. Users can query views created on top of Stripe APIs using the DBMS_CLOUD package to get Stripe information such as invoices, subscriptions, and customers.

For more information on Stripe, see the Stripe website.

- Prerequisites to Use Stripe Views
  To use Stripe with Autonomous Database, set Network ACL for accessing Stripe and create a credential that allows access to Stripe.
- STRIPE_ACCOUNTS View
  STRIPE_ACCOUNTS lists the Stripe account information for the authenticated caller.
- STRIPE_COUPONS View
  STRIPE_COUPONS lists the coupons issued in the Stripe account.

- STRIPE_CUSTOMERS View
  STRIPE_CUSTOMERS lists the customers defined in the Stripe account.
- STRIPE_INVOICES View
  STRIPE_INVOICES lists the invoices defined in the Stripe account.
- STRIPE_PLANS View
  STRIPE_PLANS lists the pricing plans defined in the Stripe account.
- STRIPE_PRODUCTS View
  STRIPE_PRODUCTS lists the query information about products defined in the Stripe account.
- STRIPE_SUBSCRIPTIONS View
  STRIPE_SUBSCRIPTIONS lists the subscriptions managed in the Stripe account.

Prerequisites to Use Stripe Views

To use Stripe with Autonomous Database, set Network ACL for accessing Stripe and create a credential that allows access to Stripe.

To use Stripe API:

1. Use the `DBMS_NETWORK_ACL_ADMIN` package to set Network ACL for accessing Stripe.

```
BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'stripe.com',
         ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                             principal_name => 'FOO',
                             principal_type => xs_acl.ptype_db)
   );
END;
/
```

2. Only an ADMIN has access to the Stripe Views. A user must be granted READ access to use them.

   For Example:

```
GRANT READ on STRIPE_COUPONS to TYLER;
```

3. Create a Credential for Stripe APIs. There are two ways to perform this task:

   - (Option 1: Create a credential object with the name `STRIPE$CRED`.) The username is `STRIPE_TOKEN` and the password is the Stripe API token. This token can be found in the Stripe Developers Dashboard. See Stripe API keys for additional information.

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'STRIPE$CRED',
        username        => 'STRIPE_TOKEN',
        password        => 'bearer_token' );
END;
/
```

   - (Option 2: Create a credential object with a user defined name.) The username is `STRIPE_TOKEN` and the password is the Stripe API token. This token can be found in the Stripe Developers Dashboard. Before selecting from the Stripe Views, the Session

parameter `DEFAULT_CREDENTIAL` should be set using the owner and credential object name. See Stripe API keys for additional information.

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'MY_STRIPE_CRED',
        username        => 'STRIPE_TOKEN',
        password        => 'bearer_token');
END;
/
ALTER SESSION SET default_credential = 'MY_SCHEMA.MY_STRIPE_CRED';
```

4. Test by querying a Stripe View:

```
SELECT name, percent_off, duration, times_redeemed FROM STRIPE_COUPONS;

NAME              PERCENT_OFF DURATION        TIMES_REDEEMED
--------------- ------------- --------------- --------------
Promotion Feb           10.00 once                         0
Seasonal disc            5.00 once                         0
Firstpurchase            5.00 forever                      1
```

STRIPE_ACCOUNTS View

STRIPE_ACCOUNTS lists the Stripe account information for the authenticated caller.

See Account for additional information.

Column	Datatype	Description
ID	VARCHAR2(4000)	Unique identifier for the object.
OBJECT	VARCHAR2(4000)	String representing the objects type.
BUSINESS_TYPE	VARCHAR2(4000)	The business type.
BUSINESS_PROFILE	CLOB	Business information about the account.
CAPABILITIES	CLOB	A hash containing the set of capabilities that was requested for this account and their associated states.
CHARGES_ENABLED	VARCHAR2(4000)	Whether the account can create live charges.
COMPANY	CLOB	Information about the company or business.
CONTROLLER	CLOB	The controller of the account.
COUNTRY	VARCHAR2(4000)	The account country.
CREATED	VARCHAR2(4000)	Time at which the account was connected.
DEFAULT_CURRENCY	VARCHAR2(4000)	Three-letter ISO currency code representing the default currency for the account.
DETAILS_SUBMITTED	VARCHAR2(4000)	Whether account details have been submitted.
EMAIL	VARCHAR2(4000)	An email address associated with the account.
EXTERNAL_ACCOUNTS	CLOB	External accounts currently attached to this account.
FUTURE_REQUIREMENTS	VARCHAR2(4000)	Information about the upcoming new requirements.
INDIVIDUAL	CLOB	Information about the person represented by the account.

Column	Datatype	Description
METADATA	VARCHAR2(4000)	Set of key-value pairs that you can attach to an object.
PAYOUTS_ENABLED	VARCHAR2(4000)	Whether Stripe can send payouts to this account.
REQUIREMENTS	CLOB	Information about the requirements for the account.
SETTINGS	CLOB	Options for customizing how the account functions within Stripe.
TOS_ACCEPTANCE	CLOB	Details on the acceptance of the Stripe Services Agreement.
TYPE	VARCHAR2(4000)	The Stripe account type.

STRIPE_COUPONS View

STRIPE_COUPONS lists the coupons issued in the Stripe account.

See Discounts for subscriptions for additional information.

Column	Datatype	Description
ID	VARCHAR2(4000)	Unique identifier for the object.
OBJECT	VARCHAR2(4000)	String representing the object type.
AMOUNT_OFF	NUMBER	Amount that will be taken off the subtotal of any invoices for this customer.
CREATED	VARCHAR2(4000)	Time at which the object was created.
CURRENCY	VARCHAR2(4000)	If amount_off has been set, the three-letter ISO code for the currency of the amount to take off.
DURATION	VARCHAR2(4000)	Describes how long a customer who applies this coupon will get the discount.
DURATION_IN_MONTHS	NUMBER	If duration is repeating, the number of months the coupon applies.
LIVEMODE	VARCHAR2(4000)	Has the value true if the object exists in live mode or the value false if the object exists in test mode.
MAX_REDEMPTIONS	NUMBER	Maximum number of times this coupon can be redeemed, in total, across all customers, before it is no longer valid.
METADATA	CLOB	Set of key-value pairs that you can attach to an object.
NAME	VARCHAR2(4000)	Name of the coupon displayed to customers on for instance invoices or receipts.

Column	Datatype	Description
PERCENT_OFF	NUMBER	Percent that will be taken off the subtotal of any invoices for this customer for the duration of the coupon.
REDEEM_BY	VARCHAR2(4000)	Date after which the coupon can no longer be redeemed.
TIMES_REDEEMED	NUMBER	Number of times this coupon has been applied to a customer.
VALID	VARCHAR2(4000)	Taking account of the above properties, whether this coupon can still be applied to a customer.

STRIPE_CUSTOMERS View

STRIPE_CUSTOMERS lists the customers defined in the Stripe account.

See Customers for additional information.

Column	Datatype	Description
ID	VARCHAR2(4000)	Unique identifier for the customer.
OBJECT	VARCHAR2(4000)	String representing the object type.
ADDRESS	CLOB	The customers address.
BALANCE	NUMBER	Current balance, if any, being stored on the customer.
CREATED	VARCHAR2(4000)	Time at which the object was created.
CURRENCY	VARCHAR2(4000)	Three-letter ISO code for the currency the customer can be charged in for recurring billing purposes.
DEFAULT_SOURCE	VARCHAR2(4000)	ID of the default payment source for the customer.
DELETED	VARCHAR2(4000)	True if the customer is marked as deleted.
DELINQUENT	VARCHAR2(4000)	When the customers latest invoice is billed by charging automatically, delinquent is true if the invoices latest charge failed.
DESCRIPTION	VARCHAR2(4000)	An arbitrary string attached to the object.
DISCOUNT	CLOB	Describes the current discount active on the customer, if there is one.
EMAIL	VARCHAR2(4000)	The customer email address.
INVOICE_PREFIX	VARCHAR2(4000)	The prefix for the customer used to generate unique invoice numbers.
INVOICE_SETTINGS	CLOB	The customer default invoice settings.

Column	Datatype	Description
LIVEMODE	VARCHAR2(4000)	Has the value true if the object exists in live mode or then value false if the object exists in test mode.
METADATA	CLOB	Set of key-value pairs that you can attach to an object.
NAME	VARCHAR2(4000)	The customer full name or business name.
NEXT_INVOICE_SEQUENCE	NUMBER	The suffix of the customers next invoice number.
PHONE	VARCHAR2(4000)	The customers phone number.
PREFERRED_LOCALES	CLOB	The customers preferred locales, ordered by preference.
SHIPPING	CLOB	Mailing and shipping address for the customer.
TAX_EXEMPT	VARCHAR2(4000)	Describes the customers tax exemption status.
TAX_IDS	CLOB	The customer tax IDs.
TEST_CLOCK	CLOB	ID of the test clock this customer belongs to.

STRIPE_INVOICES View

STRIPE_INVOICES lists the invoices defined in the Stripe account.

See Invoicing for additional information.

Column	Datatype	Description
ID	VARCHAR2(4000)	Unique identifier for the object
OBJECT	VARCHAR2(4000)	String representing the objects type.
ACCOUNT_COUNTRY	VARCHAR2(4000)	The country of the business associated with this invoice.
ACCOUNT_NAME	VARCHAR2(4000)	The public name of the business associated with this invoice.
ACCOUNT_TAX_IDS	CLOB	The account tax IDs associated with the invoice.
AMOUNT_DUE	NUMBER	Final amount due at this time for this invoice.
AMOUNT_PAID	NUMBER	The amount, in cents, that was paid.
AMOUNT_REMAINING	NUMBER	The difference between amount_due and amount_paid, in cents.
AMOUNT_SHIPPING	NUMBER	This is the sum of all the shipping amounts.
APPLICATION	VARCHAR2(4000)	ID of the Connect Application that created the invoice.

Column	Datatype	Description
APPLICATION_FEE_AMOUNT	NUMBER	The fee in cents that will be applied to the invoice and transferred to the application owners Stripe account when the invoice is paid.
ATTEMPT_COUNT	NUMBER	Number of payment attempts made for this invoice, from the perspective of the payment retry schedule.
ATTEMPTED	VARCHAR2(4000)	Whether an attempt has been made to pay the invoice.
AUTO_ADVANCE	VARCHAR2(4000)	Controls whether Stripe will perform automatic collection of the invoice.
AUTOMATIC_TAX	CLOB	Settings and latest results for automatic tax lookup for this invoice.
BILLING_REASON	VARCHAR2(4000)	Indicates the reason why the invoice was created.
CHARGE	CLOB	ID of the latest charge generated for this invoice, if any.
COLLECTION_METHOD	VARCHAR2(4000)	Either charge_automatically, or send_invoice.
CREATED	VARCHAR2(4000)	Time at which the object was created.
CURRENCY	VARCHAR2(4000)	Three-letter ISO currency code, in lowercase.
CUSTOM_FIELDS	CLOB	Custom fields displayed on the invoice.
CUSTOMER	VARCHAR2(4000)	The ID of the customer who will be billed.
CUSTOMER_ADDRESS	CLOB	The customers address.
CUSTOMER_EMAIL	VARCHAR2(4000)	The customers email.
CUSTOMER_NAME	VARCHAR2(4000)	The customers name.
CUSTOMER_PHONE	VARCHAR2(4000)	The customers phone number.
CUSTOMER_SHIPPING	CLOB	The customers shipping information.
CUSTOMER_TAX_EXEMPT	VARCHAR2(4000)	The customers tax exempt status.
CUSTOMER_TAX_IDS	CLOB	The customers tax IDs.
DEFAULT_PAYMENT_METHOD	VARCHAR2(4000)	ID of the default payment method for the invoice.
DEFAULT_SOURCE	VARCHAR2(4000)	ID of the default payment source for the invoice.
DEFAULT_TAX_RATES	CLOB	The tax rates applied to this invoice, if any.
DESCRIPTION	VARCHAR2(4000)	An arbitrary string attached to the object.
DISCOUNT	CLOB	Describes the current discount applied to this invoice, if there is one.

Column	Datatype	Description
DISCOUNTS	CLOB	The discounts applied to the invoice.
DUE_DATE	VARCHAR2(4000)	The date on which payment for this invoice is due.
ENDING_BALANCE	NUMBER	Ending customer balance after the invoice is finalized.
FOOTER	VARCHAR2(4000)	Footer displayed on the invoice.
FROM_INVOICE	VARCHAR2(4000)	Details of the invoice that was cloned.
HOSTED_INVOICE_URL	VARCHAR2(4000)	The URL for the hosted invoice page.
INVOICE_PDF	VARCHAR2(4000)	The link to download the PDF for the invoice.
LAST_FINALIZATION_ERROR	CLOB	The error encountered during the previous attempt to finalize the invoice.
LATEST_REVISION	VARCHAR2(4000)	The ID of the most recent non-draft revision of this invoice.
LINES	CLOB	The individual line items that make up the invoice.
LIVEMODE	VARCHAR2(4000)	Has the value true if the object exists in live mode or the value false if the object exists in test mode.
METADATA	CLOB	Set of key-value pairs that you can attach to an object.
NEXT_PAYMENT_ATTEMPT	VARCHAR2(4000)	The time at which payment will next be attempted.
NUMBER_	VARCHAR2(4000)	A unique, identifying string that appears on emails sent to the customer for this invoice.
ON_BEHALF_OF	VARCHAR2(4000)	The account (if any) for which the funds of the invoice payment are intended.
PAID	VARCHAR2(4000)	Whether payment was successfully collected for this invoice.
PAID_OUT_OF_BAND	VARCHAR2(4000)	Returns true if the invoice was manually marked paid, returns false if the invoice hasn't been paid yet or was paid on Stripe.
PAYMENT_INTENT	CLOB	The PaymentIntent associated with this invoice.
PAYMENT_SETTINGS	CLOB	Configuration settings for the PaymentIntent that is generated when the invoice is finalized.
PERIOD_END	VARCHAR2(4000)	End of the usage period during which invoice items were added to this invoice.
PERIOD_START	VARCHAR2(4000)	Start of the usage period during which invoice items were added to this invoice.

Column	Datatype	Description
POST_PAYMENT_CREDIT_NOTES_AMOUNT	NUMBER	Total amount of all post-payment credit notes issued for this invoice.
PRE_PAYMENT_CREDIT_NOTES_AMOUNT	NUMBER	Total amount of all pre-payment credit notes issued for this invoice.
QUOTE	VARCHAR2(4000)	The quote this invoice was generated from.
RECEIPT_NUMBER	VARCHAR2(4000)	This is the transaction number that appears on email receipts sent for this invoice.
RENDERING_OPTIONS	CLOB	Options for invoice PDF rendering.
SHIPPING_COST	CLOB	The details of the cost of shipping, including the ShippingRate applied on the invoice.
SHIPPING_DETAILS	CLOB	Shipping details for the invoice.
STARTING_BALANCE	NUMBER	Starting customer balance before the invoice is finalized.
STATEMENT_DESCRIPTOR	VARCHAR2(4000)	Extra information about an invoice for the customers credit card statement.
STATUS	VARCHAR2(4000)	The status of the invoice, one of draft, open, paid, uncollectible, or void.
STATUS_TRANSITIONS	CLOB	The timestamps at which the invoice status was updated.
SUBSCRIPTION	VARCHAR2(4000)	The subscription that this invoice was prepared for, if any.
SUBTOTAL	NUMBER	Total of all subscriptions, invoice items, and prorations on the invoice before any invoice level discount or exclusive tax is applied.
SUBTOTAL_EXCLUDING_TAX	NUMBER	The integer amount in cents representing the subtotal of the invoice before any invoice level discount or tax is applied.
TAX	NUMBER	The amount of tax on this invoice.
THRESHOLD_REASON	CLOB	If billing_reason is set to subscription_threshold this returns more information on which threshold rules triggered the invoice.
TEST_CLOCK	VARCHAR2(4000)	ID of the test clock this invoice belongs to.
TOTAL	NUMBER	Total after discounts and taxes.
TOTAL_DISCOUNT_AMOUNTS	CLOB	The aggregate amounts calculated per discount across all line items.

Column	Datatype	Description
TOTAL_EXCLUDING_TAX	NUMBER	The integer amount in cents representing the total amount of the invoice including all discounts but excluding all tax.
TOTAL_TAX_AMOUNTS	CLOB	The aggregate amounts calculated per tax rate for all line items.
TRANSFER_DATA	CLOB	The account (if any) the payment will be attributed to for tax reporting, and where funds from the payment will be transferred to for the invoice.
WEBHOOKS_DELIVERED_AT	VARCHAR2(4000)	This field tracks the time when webhooks for this invoice were successfully delivered.

STRIPE_PLANS View

STRIPE_PLANS lists the pricing plans defined in the Stripe account.

See Plans for additional information.

Column	Datatype	Description
ID	VARCHAR2(4000)	Unique identifier for the plan.
OBJECT	VARCHAR2(4000)	String representing the object type.
ACTIVE	VARCHAR2(4000)	Whether the plan is currently available for purchase.
AGGREGATE_USAGE	VARCHAR2(4000)	Specifies a usage aggregation strategy for plans of usage_type=metered.
AMOUNT	NUMBER	The unit amount in cents to be charged, represented as a whole integer.
AMOUNT_DECIMAL	VARCHAR2(4000)	The unit amount in cents to be charged, represented as a decimal string with at most 12 decimal places.
BILLING_SCHEME	VARCHAR2(4000)	Describes how to compute the price per period.
CREATED	VARCHAR2(4000)	Time at which the plan was created.
CURRENCY	VARCHAR2(4000)	Three-letter ISO currency code, in lowercase.
DELETED	VARCHAR2(4000)	True if the plan is marked as deleted.
INTERVAL	VARCHAR2(4000)	The frequency at which a subscription is billed.
INTERVAL_COUNT	NUMBER	The number of intervals between subscription billings.

Column	Datatype	Description
LIVEMODE	VARCHAR2(4000)	Has the value true if the plan exists in live mode or the value false if the plan exists in test mode.
METADATA	CLOB	Set of key-value pairs that you can attach to an plan.
NICKNAME	VARCHAR2(4000)	A brief description of the plan, hidden from customers.
PRODUCT	VARCHAR2(4000)	ID of the product whose pricing this plan determines.
TIERS	CLOB	Each element represents a pricing tier.
TIERS_MODE	VARCHAR2(4000)	Defines if the tiering price should be graduated or volume based.
TRANSFORM_USAGE	CLOB	Apply a transformation to the reported usage or set quantity before computing the amount billed.
TRIAL_PERIOD_DAYS	NUMBER	Default number of trial days when subscribing a customer to this plan using trial_from_plan=true.
USAGE_TYPE	VARCHAR2(4000)	Configures how the quantity per period should be determined.

STRIPE_PRODUCTS View

STRIPE_PRODUCTS lists the query information about products defined in the Stripe account.

See Manage products and prices for additional information.

Column	Datatype	Description
ID	VARCHAR2(4000)	Unique identifier for the object.
OBJECT	VARCHAR2(4000)	String representing the object type.
ACTIVE	VARCHAR2(4000)	Whether the product is currently available for purchase.
CREATED	VARCHAR2(4000)	Time at which the object was created.
DEFAULT_PRICE	VARCHAR2(4000)	The ID of the Price object that is the default price for this product.
DESCRIPTION	VARCHAR2(4000)	The product description, meant to be displayable to the customer.
IMAGES	CLOB	A list of up to 8 URLs of images for this product, meant to be displayable to the customer.
LIVEMODE	VARCHAR2(4000)	Has the value true if the object exists in live mode or the value false if the object exists in test mode.
METADATA	CLOB	Set of key-value pairs that you can attach to an object.

Column	Datatype	Description
NAME	VARCHAR2(4000)	The product name, meant to be displayable to the customer.
PACKAGE_DIMENSIONS	CLOB	The dimensions of this product for shipping purposes.
SHIPPABLE	VARCHAR2(4000)	Whether this product is shipped.
STATEMENT_DESCRIPTOR	VARCHAR2(4000)	Extra information about a product which will appear on your customers credit card statement.
TAX_CODE	VARCHAR2(4000)	A tax code ID.
UNIT_LABEL	VARCHAR2(4000)	A label that represents units of this product.
UPDATED	VARCHAR2(4000)	Time at which the object was last updated.
URL	VARCHAR2(4000)	A URL of a publicly-accessible webpage for this product.

STRIPE_SUBSCRIPTIONS View

STRIPE_SUBSCRIPTIONS lists the subscriptions managed in the Stripe account.

See Subscriptions for additional information.

Column	Datatype	Subscription
ID	VARCHAR2(4000)	Unique identifier for the object.
OBJECT	VARCHAR2(4000)	String representing the object type.
APPLICATION	VARCHAR2(4000)	ID of the Connect Application that created the subscription.
APPLICATION_FEE_PERCENT	NUMBER	A non-negative decimal between 0 and 100, with at most two decimal places.
AUTOMATIC_TAX	CLOB	Automatic tax settings for this subscription.
BILLING_CYCLE_ANCHOR	VARCHAR2(4000)	Determines the date of the first full invoice, and, for plans with month or year intervals, the day of the month for subsequent invoices.
BILLING_THRESHOLDS	CLOB	Define thresholds at which an invoice will be sent, and the subscription advanced to a new billing period.
CANCEL_AT	VARCHAR2(4000)	A date in the future at which the subscription will automatically get canceled.
CANCEL_AT_PERIOD_END	VARCHAR2(4000)	If the subscription has been canceled with the at_period_end flag set to true, cancel_at_period_end on the subscription will be true.

Column	Datatype	Subscription
CANCELED_AT	VARCHAR2(4000)	If the subscription has been canceled, the date of that cancellation.
COLLECTION_METHOD	VARCHAR2(4000)	Either charge_automatically, or send_invoice.
CREATED	VARCHAR2(4000)	Time at which the object was created.
CURRENCY	VARCHAR2(4000)	Three-letter ISO currency code, in lowercase.
CURRENT_PERIOD_END	VARCHAR2(4000)	End of the current period that the subscription has been invoiced for.
CURRENT_PERIOD_START	VARCHAR2(4000)	Start of the current period that the subscription has been invoiced for.
CUSTOMER	VARCHAR2(4000)	ID of the customer who owns the subscription.
DAYS_UNTIL_DUE	NUMBER	Number of days a customer has to pay invoices generated by this subscription.
DEFAULT_PAYMENT_METHOD	VARCHAR2(4000)	ID of the default payment method for the subscription.
DEFAULT_SOURCE	VARCHAR2(4000)	ID of the default payment source for the subscription.
DEFAULT_TAX_RATES	CLOB	The tax rates that will apply to any subscription item that does not have tax_rates set.
DESCRIPTION	VARCHAR2(4000)	The subscription description, meant to be displayable to the customer.
DISCOUNT	CLOB	Describes the current discount applied to this subscription, if there is one.
ENDED_AT	VARCHAR2(4000)	If the subscription has ended, the date the subscription ended.
ITEMS	CLOB	List of subscription items, each with an attached price.
LATEST_INVOICE	VARCHAR2(4000)	The most recent invoice this subscription has generated.
LIVEMODE	VARCHAR2(4000)	Has the value true if the object exists in live mode or the value false if the object exists in test mode.
METADATA	CLOB	Set of key-value pairs that you can attach to an object.
NEXT_PENDING_INVOICE_ITEM_INVOICE	VARCHAR2(4000)	Specifies the approximate timestamp on which any pending invoice items will be billed acc to the schedule provided at pending_invoice_item_interval.
ON_BEHALF_OF	VARCHAR2(4000)	The account the charge was made on behalf of for charges associated with this subscription.

Column	Datatype	Subscription
PAUSE_COLLECTION	CLOB	If specified, payment collection for this subscription will be paused.
PAYMENT_SETTINGS	VARCHAR2(4000)	Payment settings passed on to invoices created by the subscription.
PENDING_INVOICE_ITEM_INTERVAL	CLOB	Specifies an interval for how often to bill for any pending invoice items.
PENDING_SETUP_INTENT	VARCHAR2(4000)	For collecting user authentication when creating a subscription without immediate payment or updating a subscriptions payment method, allowing you to optimize for off-session payments.
PENDING_UPDATE	CLOB	If specified, pending updates that will be applied to the subscription once the latest_invoice has been paid.
SCHEDULE	CLOB	The schedule attached to the subscription.
START_DATE	VARCHAR2(4000)	Date when the subscription was first created.
STATUS	VARCHAR2(4000)	Possible values are incomplete, incomplete_expired, trialing, active, past_due, canceled, or unpaid.
TEST_CLOCK	VARCHAR2(4000)	ID of the test clock this subscription belongs to.
TRANSFER_DATA	CLOB	The account the subscriptions payments will be attributed to for tax reporting, and where funds from each payment will be transferred to for each of the subscriptions invoices.
TRIAL_END	VARCHAR2(4000)	If the subscription has a trial, the end of that trial.
TRIAL_SETTINGS	VARCHAR2(4000)	Settings related to subscription trials.
TRIAL_START	VARCHAR2(4000)	If the subscription has a trial, the beginning of that trial.

Autonomous Database – Oracle Database Features

Describes Oracle Database features available with Autonomous Database.

Autonomous Database includes features that:

- Automate index management tasks, such as creating, rebuilding, and dropping indexes based on changes in the application workload.

  See Manage Automatic Indexing on Autonomous Database for more information.

> **Note:**
>
> There are restrictions for Automatic Indexing when you use JSON data with Autonomous Database. See SODA Notes for more information.

- Gather real-time statistics automatically while a conventional DML workload is running. Because statistics can go stale between stats gathering jobs, online statistics gathering for conventional DML helps the optimizer generate more optimal plans. Online statistics aim to reduce the possibility of the optimizer being misled by stale statistics.

  See Real-Time Statistics for more information.

- Gather statistics automatically on a more frequent basis. High-Frequency Automatic Optimizer Statistics Collection complements the standard statistics collection job. By default, the collection occurs every 15 minutes, meaning that statistics have less time in which to be stale. High-Frequency Automatic Optimizer Statistics Collection is enabled by default.

  See Configuring High-Frequency Automatic Optimizer Statistics Collection for more information.

- Quarantine execution plans for SQL statements, for example, statements that are terminated by the Resource Manager for consuming excessive system resources in an Oracle Database. Automatic SQL Quarantine based on Resource Manager consumption limit violations is disabled by default but any manually quarantined SQL statement will be honored.

  See Quarantine for Execution Plans for SQL Statements Consuming Excessive System Resources for more information.

- Automatically assess the opportunity for SQL plan changes to improve the performance for known statements.

  See Managing the SPM Evolve Advisor Task for more information.

- Apache ORC format is supported in Autonomous Database for loading and querying data in object store.

  See Create Credentials and Load Data Pump Dump Files into an Existing Table and Query External Data with ORC, Parquet, or Avro Source Files for more information.

- Complex types are supported in Autonomous Database for ORC, Avro, and Parquet structured files.

  See DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types for more information.

Always Free Autonomous Database – Oracle Database 21c

When you provision Always Free Autonomous Database you can select either Oracle Database 19c or Oracle Database 23ai.

- Always Free Autonomous Database Oracle Database 21c Features
  When you provision Always Free Autonomous Database you can select either Oracle Database 19c or Oracle Database 23ai.
- Always Free Autonomous Database Oracle Database 21c Notes
  If you are using Always Free Autonomous Database with Oracle Database 21c, the following Oracle Database 21c functionality is not supported:

Always Free Autonomous Database Oracle Database 21c Features

When you provision Always Free Autonomous Database you can select either Oracle Database 19c or Oracle Database 23ai.

> **Note:**
>
> With the availability of Always Free Autonomous Database Oracle Database 23ai, Oracle Database 21c is no longer available as a provisioning or cloning option. Existing Always Free Autonomous Databases running with Oracle Database 21c continue as Always Free Autonomous Database.

Always Free Autonomous Database running with Oracle Database 21c offers many new innovative autonomous and developer-oriented functionality, including but not limited to the following:

Performance Features

- **Automatic Zone Maps**

  Automatic zone maps are created and maintained for any user table without any customer intervention. Zone maps allow the pruning of block ranges and partitions based on the predicates in the queries. Automatic zone maps are maintained for direct loads, and are maintained and refreshed for any other DML operation incrementally and periodically in the background.

  The feature is enabled as follows:

  ```
  exec dbms_auto_zonemap.configure('AUTO_ZONEMAP_MODE','ON');
  ```

  The feature is disabled as follows:

  ```
  exec dbms_auto_zonemap.configure('AUTO_ZONEMAP_MODE','OFF');
  ```

  See Summary of DBMS_AUTO_ZONEMAP Subprograms for more information.

- **Object Activity Tracking System**

  Object Activity Tracking System (OATS) tracks the usage of various types of database objects. Usage includes operations such as access, data manipulation, or refresh.

  No manual intervention is required to enable OATS, and zero or minimal configuration is required. See PL/SQL procedure DBMS_ACTIVITY.CONFIGURE and database dictionary views DBA_ACTIVITY_CONFIG for details.

Application Development: Advanced Analytical SQL Capabilities

- **SQL Macros**

  SQL Macros, the capability to factor out common SQL constructs supports scalar expressions, increasing developer productivity, simplify collaborative code development, and improve code quality. See SQL Macros for more information.

- **Enhanced Analytic Functions**

Window functions support the full ANSI Standard, including the support of EXCLUDE options and the WINDOW clause. Supporting the full ANSI standard enables easier migration of applications that were developed with other standard-compliant database systems. See Windowing Functions for more information.

- **New Analytical and Statistical Aggregate Functions**

  Several new analytical and statistical aggregate functions are available in SQL in Oracle Database 21c. With these additional SQL aggregation functions, you can write more efficient code and benefit from faster in-database processing.

  - `CHECKSUM` computes the checksum of the input values or expression.

    Supports the keywords `ALL` and `DISTINCT`.

  - `KURTOSIS` functions `KURTOSIS_POP` and `KURTOSIS_SAMP` measure the tailedness of a data set where a higher value means more of the variance within the data set is the result of infrequent extreme deviations as opposed to frequent modestly sized deviations. Note that a normal distribution has a kurtosis of zero.

    Supports the keywords `ALL`, `DISTINCT`, and `UNIQUE`.

  - `SKEWNESS` functions `SKEWNESS_POP` and `SKEWNESS_SAMP` are measures of asymmetry in data. A positive skewness is means the data skews to the right of the center point. A negative skewness means the data skews to the left.

    Supports the keywords `ALL`, `DISTINCT`, and `UNIQUE`.

  - `ANY_VALUE`, a function to simplify and optimize the performance of `GROUP BY` statements, returns a random value in a group and is optimized to return the first value in the group. It ensures that there are no comparisons for any incoming row and eliminates the necessity to specify every column as part of the `GROUP BY` clause.

  See Oracle Database 21c SQL Language Reference Guide for more information.

- **Bitwise Aggregate Functions**

  With the new bitwise type processing functions `BIT_AND_AGG`, `BIT_OR_AGG`, and`BIT_XOR_AGG`, native bitwise type processing is provided by Oracle Database 21c. These functions enable a type of processing inside the database for new types of application processing, improving the overall performance, avoiding unnecessary data movement, and natively taking advantage of core database functionality such as parallel processing. See Oracle Database 21c SQL Language Reference Guide. for more information.

JavaScript Execution using DBMS_MLE

The `DBMS_MLE` package allows users to execute JavaScript code inside the Oracle Database and exchange data seamlessly between PL/SQL and JavaScript. The JavaScript code itself can execute PL/SQL and SQL through built-in JavaScript modules. JavaScript data types are automatically mapped to Oracle Database data types and vice versa.

With the `DBMS_MLE` package, developers can write their data processing logic in JavaScript. JavaScript is a widely-used and popular programming language that can now also be used for writing programs that need to execute close to the data.

See DBMS_MLE for more information.

Blockchain Table

Blockchain tables are append-only tables in which only insert operations are allowed. Deleting rows is either prohibited or restricted based on time. Rows in a blockchain table are made tamper-resistant by special sequencing and chaining algorithms. Users can verify that rows

have not been tampered. A hash value that is part of the row metadata is used to chain and validate rows.

Blockchain tables enable you to implement a centralized ledger model where all participants in the blockchain network have access to the same tamper-resistant ledger.

A centralized ledger model reduces administrative overheads of setting up a decentralized ledger network, leads to a relatively lower latency compared to decentralized ledgers, enhances developer productivity, reduces the time to market, and leads to significant savings for the organization. Database users can continue to use the same tools and practices that they would use for other database application development.

See Managing Blockchain Tables for more information.

JSON Document Store Enhancements

- **Enhancements to Data Guide**

  Enhances development flexibility and allows for materialized views, which may improve query performance with a trade-off against DML performance.

  – `JSON_DATAGUIDE` now gathers statistic information if you specify `DBMS_JSON.GATHER_STATS` in the third argument. They are computed dynamically (up-to-date) at the time of the function call.

  – `DBMS_JSON.CREATE_VIEW` now gives you the option to create a materialized view instead of a standard view. It also gives you the option to specify a particular path so the view can be created on a subset of the data. Both `CREATE_VIEW` and `ADD_VIRTUAL_COLUMN` are enhanced to allow automatic resolution of column naming conflicts, to provide a prefix to be applied to column names, and to specify the case-sensitivity of column names.

  See JSON Data Guide for more information.

- **Multivalue Index for JSON DataType**

  A new create index syntax `CREATE MULTIVALUE INDEX` allows you to create a functional index on arrays of strings or numbers within a JSON datatype column. Each unique value within the array will become a searchable index entry. This avoids the need for full JSON scans to find values within arrays in JSON columns, when searched using the `JSON_EXISTS` or `JSON_VALUE` operators. It provides similar benefits to conventional functional indexes when searching JSON, but conventional functional indexes are limited to a single indexed value per row.

  See Creating Multivalue Function-Based Indexes for JSON_EXISTS and Using a Multivalue Function-Based Index for more information.

- **New JSON Data Type**

  JSON is a new SQL and PL/SQL data type for JSON data. Using this type provides a substantial increase in query and update performance. JSON data type uses binary format OSON that is optimized for SQL/JSON query and DML processing. Using the binary format can yield database performance improvements for processing JSON data.

  You can use JSON data type and its instances in most places where a SQL data type is allowed, including:

  – As the column type for table or view DDL

  – With SQL/JSON functions and conditions, and with PL/SQL procedures and functions

  – In Oracle dot-notation query syntax

  – For creation of functional and search indexes

Oracle Call Interface and Java Database Connectivity (JDBC) clients now provide APIs that can work directly with binary JSON datatype OSON format, significantly saving network costs and server CPU cycles. Going forward, Oracle recommends using JSON datatype to store and process JSON data.

The Oracle Autonomous JSON Database uses OSON format to store and process JSON data.

See Creating a Table With a JSON Column for more information.

- **New Oracle SQL Function JSON_TRANSFORM**

  You can use SQL function `JSON_TRANSFORM` to update parts of a JSON document. You specify which parts to modify, the modifications, and any new values. `JSON_TRANSFORM` is optimized by doing partial updates at OSON format level to achieve better JSON datatype update performance.

  `JSON_TRANSFORM` makes it easier for an application to modify a JSON document, without having to parse and rebuild it. In most cases, it also avoids a round-trip between the server and client for the whole document.

  See Oracle SQL Function JSON_TRANSFORM for more information.

- **SQL/JSON Syntax Improvements**

  You can now express more complex SQL/JSON queries and express some queries more succinctly:

  – New SQL function `JSON_SCALAR` accepts a scalar instance of a SQL data type and returns a scalar JSON value as an instance of JSON data type.

  – New JSON path-language item methods support `JSON_SCALAR: float()`, `double()`, `binary()`, `ymInterval()`, and `dsInterval()`.

  – The JSON path-language and dot-notation syntax support the aggregate item methods: `avg()`, `count()`, `minNumber()`, `maxNumber()`, `minString()`, `maxString()`, `sum()`.

    See Simple Dot-Notation Access to JSON Data and SQL/JSON Path Expression Item Methods for more information.

SODA Enhancements: New JSON Data Type

The default collection storage changes to the JSON data type. See Creating a Document Collection with SODA for PL/SQL for more information.

PL/SQL Enhancements

- PL/SQL is enhanced to help you program iteration controls using new iterators in loops and in qualified expressions.

  The new iterator constructs are clear, simple, understandable, and efficient.

  See PL/SQL Extended Iterators for more information.

Gradual Database Password Rollover for Applications

An application can change its database passwords without an administrator having to schedule downtime.

To accomplish this, a database administrator can associate a profile having a non-zero limit for the `PASSWORD_ROLLOVER_TIME` password profile parameter, with an application schema. This allows the database password of the application user to be altered while allowing the older

password to remain valid for the time specified by the `PASSWORD_ROLLOVER_TIME` limit. During the rollover period of time, the application instance can use either the old password or the new password to connect to the database server. When the rollover time expires, only the new password is allowed.

In addition to the clause `PASSWORD_ROLLOVER_TIME` in the `CREATE PROFILE` and `ALTER PROFILE` statements, the `ALTER USER` statement has a clause, `EXPIRE PASSWORD ROLLOVER PERIOD`. The `ACCOUNT_STATUS` column of the `DBA_USERS` and `USER_USERS` data dictionary views have several statuses indicating values to indicate rollover status.

See Managing Gradual Database Password Rollover for Applications for more information.

Always Free Autonomous Database Oracle Database 21c Notes

If you are using Always Free Autonomous Database with Oracle Database 21c, the following Oracle Database 21c functionality is not supported:

- Automatic Materialized Views
- Autonomous Database only supports Cloud Links when your database version is Oracle Database 19c. Cloud Links are not supported with database version Oracle Database 21c. See Use Cloud Links for Read Only Data Access on Autonomous Database for more information.

Always Free Autonomous Database – Oracle Database 23ai

When you provision Always Free Autonomous Database you can select either Oracle Database 19c or Oracle Database 23ai.

- Always Free Autonomous Database Oracle Database 23ai Features
  Always Free Autonomous Database with Oracle Database 23ai offers many new innovative autonomous features and developer-oriented functionality.
- Always Free Autonomous Database Oracle Database 23ai Notes
  If you are using Always Free Autonomous Database with Oracle Database 23ai, the following Oracle Database 23ai functionality is not supported:

Always Free Autonomous Database Oracle Database 23ai Features

Always Free Autonomous Database with Oracle Database 23ai offers many new innovative autonomous features and developer-oriented functionality.

If you are using Always Free Autonomous Database with Oracle Database 23ai, then many of the concepts and features of this service are further documented here:

- Oracle Database 23ai
- *Oracle Database New Features*

Always Free Autonomous Database Oracle Database 23ai Notes

If you are using Always Free Autonomous Database with Oracle Database 23ai, the following Oracle Database 23ai functionality is not supported:

- In-Memory Neighbor Graph vector indexes are not supported.

Autonomous Database RMAN Recovery Catalog

You can use Oracle Autonomous Database as a Recovery Manager (RMAN) recovery catalog. A recovery catalog is a database schema that RMAN uses to store metadata about one or more Oracle databases.

- Use Autonomous Database as an RMAN Recovery Catalog
  Recovery Manager (RMAN) recovery catalog is preinstalled in Autonomous Database in schema `RMAN$CATALOG`. The preinstalled catalog version is based on the latest version of Oracle Database and is compatible with all supported Oracle database versions.

Use Autonomous Database as an RMAN Recovery Catalog

Recovery Manager (RMAN) recovery catalog is preinstalled in Autonomous Database in schema `RMAN$CATALOG`. The preinstalled catalog version is based on the latest version of Oracle Database and is compatible with all supported Oracle database versions.

The recovery catalog contains metadata about RMAN operations for each registered target database. When RMAN is connected to a recovery catalog, RMAN obtains its metadata exclusively from the catalog.

> **Note:**
>
> Autonomous Database is not supported as an RMAN target database. An RMAN target database is an Oracle Database to which RMAN is connected with the `TARGET` keyword. A target database is a database on which RMAN is performing backup and recovery operations. See Backup and Restore Autonomous Database Instances for information on Autonomous Database backup and recovery operations.

Access to RMAN Recovery Catalog

Access to the recovery catalog is provided through predefined user `RMAN$CATALOG` with the appropriate access to the recovery catalog only. The `RMAN$CATALOG` user is locked by default.

You can either proxy to the predefined user `RMAN$CATALOG` through the ADMIN user or explicitly unlock the preinstalled schema:

- ADMIN user proxy into `RMAN$CATALOG` using ADMIN user's password:

  ```
  connect admin[rman$catalog]/password@connect_string
  ```

- ADMIN user can set a password for `RMAN$CATALOG`. Then the `RMAN$CATALOG` user can directly connect:

  ```
  connect admin/password@connect_string
  alter user rman$catalog identified by password account unlock;
  connect rman$catalog/password@connect_string
  ```

Use the RMAN Recovery Catalog

You can use the RMAN recovery catalog by connecting RMAN to the preinstalled recovery catalog. Registering a target database in the recovery catalog maintains the database's records in the recovery catalog. For example, to register a target database:

```
RMAN> connect catalog rman$catalog/password@connect_string;

connected to recovery catalog database
recovery catalog schema version 21.01.00.00. is newer than RMAN version

RMAN> register database;
database registered in recovery catalog
starting full resync of recovery catalog
```

To use your Autonomous Database as a recovery catalog, it is recommended to connect with the `LOW` service.

See Registering a Database in the Recovery Catalog for more details about using the RMAN recovery catalog.

Notes for Users Migrating from Other Oracle Databases

Describes information that is useful when you are migrating from other Oracle Databases to Oracle Autonomous Database.

- Initialization Parameters
  Autonomous Database configures database initialization parameters automatically when you provision a database. You do not need to set any initialization parameters to start using your service. But, you can modify some parameters if you need to.
- SQL Commands
  Autonomous Database allows most of the SQL commands available in Oracle Database. To ensure the security and the performance of Autonomous Database, some SQL commands are restricted.
- Data Types
  Autonomous Database allows most of the data types available in Oracle Database. To ensure the security and the performance of Autonomous Database, some data types are restricted.
- PL/SQL Packages Notes for Autonomous Database
  Notes for Oracle Database PL/SQL packages in Autonomous Database.
- Oracle XML DB
  Describes Autonomous Database support for Oracle XML DB features. To ensure the security and the performance of your Autonomous Database, some Oracle XML DB features are restricted.
- Oracle Text
  Describes Autonomous Database support for Oracle Text features. To ensure the security and the performance of your Autonomous Database, some Oracle Text features are restricted.
- Oracle Flashback
  Oracle Flashback Technology is a group of Oracle Database features that let you view past states of database objects or to return database objects to a previous state without using point-in-time media recovery.

- Oracle Database Real Application Security
  Oracle Database Real Application Security is a database authorization model that: supports declarative security policies, enables end-to-end security for multitier applications, provides an integrated solution to secure database and application resources, and advances the security architecture of Oracle Database to meet existing and emerging demands of applications developed for the Internet.
- Oracle LogMiner
  Describes restrictions for Oracle LogMiner on Autonomous Database.
- Choose a Character Set for Autonomous Database

Initialization Parameters

Autonomous Database configures database initialization parameters automatically when you provision a database. You do not need to set any initialization parameters to start using your service. But, you can modify some parameters if you need to.

List of Initialization Parameters that can be Modified

```
APPROX_FOR_AGGREGATION
APPROX_FOR_COUNT_DISTINCT
APPROX_FOR_PERCENTILE
BLANK_TRIMMING (Allowed only with ALTER SYSTEM)
CLIENT_PREFETCH_ROWS (see CLIENT_PREFETCH_ROWS
CONSTRAINTS (Allowed only with ALTER SESSION)
CONTAINER (Allowed only with ALTER SESSION)
CONTAINER_DATA
CURRENT_SCHEMA (Allowed only with ALTER SESSION)
CURSOR_INVALIDATION (Allowed only with ALTER SESSION)
CURSOR_SHARING
DDL_LOCK_TIMEOUT
DEFAULT_COLLATION (Allowed only with ALTER SESSION)
DEFAULT_CREDENTIAL (Allowed only with ALTER SESSION)
EDITION (Allowed only with ALTER SESSION)
FIXED_DATE (Allowed only with ALTER SYSTEM)
IGNORE_SESSION_SET_PARAM_ERRORS
ISOLATION_LEVEL (Allowed only with ALTER SESSION)
JOB_QUEUE_PROCESSES (Allowed only with ALTER SYSTEM)
LDAP_DIRECTORY_ACCESS
LOAD_WITHOUT_COMPILE
MAX_IDLE_TIME (Allowed only with ALTER SYSTEM)
MAX_STRING_SIZE (See Data Types for details)
NLS_CALENDAR
NLS_COMP
NLS_CURRENCY
NLS_DATE_FORMAT
NLS_DATE_LANGUAGE
NLS_DUAL_CURRENCY
NLS_ISO_CURRENCY
NLS_LANGUAGE
NLS_LENGTH_SEMANTICS
NLS_NCHAR_CONV_EXCP
NLS_NUMERIC_CHARACTERS
NLS_SORT
NLS_TERRITORY
NLS_TIME_FORMAT
NLS_TIME_TZ_FORMAT
NLS_TIMESTAMP_FORMAT
NLS_TIMESTAMP_TZ_FORMAT
```

```
OPTIMIZER_CAPTURE_SQL_QUARANTINE
OPTIMIZER_IGNORE_HINTS
OPTIMIZER_IGNORE_PARALLEL_HINTS
OPTIMIZER_MODE
OPTIMIZER_REAL_TIME_STATISTICS
OPTIMIZER_USE_SQL_QUARANTINE
PLSCOPE_SETTINGS
PLSQL_CCFLAGS
PLSQL_DEBUG
PLSQL_OPTIMIZE_LEVEL
PLSQL_WARNINGS
QUERY_REWRITE_INTEGRITY
READ_ONLY (Allowed only with ALTER SESSION)
RECYCLE_BIN
REMOTE_DEPENDENCIES_MODE
RESULT_CACHE_INTEGRITY (See RESULT_CACHE_INTEGRITY)
RESULT_CACHE_MODE (See RESULT_CACHE_MODE)
SESSION_EXIT_ON_PACKAGE_STATE_ERROR (Allowed only with ALTER SYSTEM) See
SESSION_EXIT_ON_PACKAGE_STATE_ERROR
SKIP_UNUSABLE_INEDEXES
SQL_TRACE (Allowed only with ALTER SESSION) See Perform SQL Tracing on Autonomous
Database for details
SQL_TRANSLATION_PROFILE (Allowed only with ALTER SESSION)
STATISTICS_LEVEL (Allowed only with ALTER SESSION)
SYSDATE_AT_DBTIMEZONE (See SYSDATE_AT_DBTIMEZONE Select a
    Time Zone for SYSDATE on Autonomous Database)
TIMEZONE (Allowed only with ALTER SESSION)
```

For more information on initialization parameters see *Oracle Database Reference*. For more information on `TIME_ZONE`, see *Oracle Database SQL Language Reference*.

For more information on `OPTIMIZER_IGNORE_HINTS` and `OPTIMIZER_IGNORE_PARALLEL_HINTS`, see Manage Optimizer Statistics on Autonomous Database.

- SESSION_EXIT_ON_PACKAGE_STATE_ERROR
  `SESSION_EXIT_ON_PACKAGE_STATE_ERROR` enables or disables special handling for stateful PL/SQL packages running in a session.
- SYSDATE_AT_DBTIMEZONE Select a Time Zone for SYSDATE on Autonomous Database
- CLIENT_PREFETCH_ROWS
- JOB_QUEUE_PROCESSES
- RESULT_CACHE_INTEGRITY
  Set the `RESULT_CACHE_INTEGRITY` parameter to specify whether the result cache considers queries using possibly non-deterministic constructs as candidates for result caching.
- RESULT_CACHE_MODE
  Set the `RESULT_CACHE_MODE` parameter to specify which queries are eligible to store result sets in the result cache. Only query execution plans with the result cache operator will attempt to read from or write to the result cache.

SESSION_EXIT_ON_PACKAGE_STATE_ERROR

`SESSION_EXIT_ON_PACKAGE_STATE_ERROR` enables or disables special handling for stateful PL/SQL packages running in a session.

Property	Description
Parameter type	Boolean
Default Value	FALSE
Modifiable	ALTER SYSTEM
Range of values	TRUE | FALSE

SESSION_EXIT_ON_PACKAGE_STATE_ERROR specifies the handling for a stateful PL/SQL package running in a session. When such a package undergoes modification, such as during planned maintenance for Oracle-supplied objects, the sessions that have an active instantiation of the package receive the following error when they attempt to run the package:

```
ORA-4068 existing state of package has been discarded
```

However, the application code that receives the ORA-4068 error may not be equipped to handle this error with its retry logic.

Setting SESSION_EXIT_ON_PACKAGE_STATE_ERROR to TRUE provides different handling for this case. When SESSION_EXIT_ON_PACKAGE_STATE_ERROR is TRUE, instead of just raising the ORA-4068 error when the package state is discarded, the session immediately exits. This can be advantageous because many applications are able to handle session termination by automatically and transparently re-establishing the connection.

SYSDATE_AT_DBTIMEZONE Select a Time Zone for SYSDATE on Autonomous Database

SYSDATE_AT_DBTIMEZONE enables special handling in a session for the date and time value returned in calls to SYSDATE and SYSTIMESTAMP.

Depending on the value of SYSDATE_AT_DBTIMEZONE, you see either the date and time based on the default Autonomous Database time zone, Coordinated Universal Time (UTC), or based on the time zone that you set in your database.

Property	Description
Parameter type	Boolean
Default Value	FALSE
Modifiable	ALTER SESSION, ALTER SYSTEM
Range of values	TRUE | FALSE

Default Autonomous Database Time Zone

The default Autonomous Database time zone is Coordinated Universal Time (UTC) and by default calls to SYSDATE and SYSTIMESTAMP return the date and time in UTC.

In order to change database time zone, you can run the following statement. This example sets the database time zone to UTC-5.

```
ALTER DATABASE SET TIME_ZONE='-05:00';
```

> **Note:**
>
> You must restart the Autonomous Database instance for the change to take effect.

After you set the database time zone, by default SYSDATE and SYSTIMESTAMP continue to return date and time in UTC (SYSDATE_AT_DBTIMEZONE is FALSE by default). If you set SYSDATE_AT_DBTIMEZONE to TRUE in a session, SYSDATE and SYSTIMESTAMP return the database time zone.

See Setting the Database Time Zone for more information on using the SET TIME_ZONE clause with ALTER DATABASE.

Using SYSDATE_AT_DBTIMEZONE in a Session

When SYSDATE_AT_DBTIMEZONE is FALSE in a session, calls to SYSDATE and SYSTIMESTAMP return values based on the default Autonomous Database time zone, Coordinated Universal Time (UTC).

When SYSDATE_AT_DBTIMEZONE is TRUE in a session, calls to SYSDATE or SYSTIMESTAMP return the date and time based on the database time zone.

> **Note:**
>
> Setting SYSDATE_AT_DBTIMEZONE to TRUE only affects the use of SYSDATE and SYSTIMESTAMP as operators in application SQL (for example, in queries, DML, and CTAS operations). When using this parameter, it is recommended that your client/session timezone matches your database timezone.

Example

The following example returns dates and times for two different time zones, based on the SYSDATE_AT_DBTIMEZONE parameter value:

```
SQL> SELECT DBTIMEZONE FROM DUAL;

DBTIMEZONE
_____________
-05:00

SQL> ALTER SESSION SET SYSDATE_AT_DBTIMEZONE=FALSE;

Session altered.

SQL> SELECT SYSTIMESTAMP FROM DUAL;

SYSTIMESTAMP
_____________________________________
27-JAN-22 06.59.45.708082000 PM GMT

SQL> ALTER SESSION SET SYSDATE_AT_DBTIMEZONE=TRUE;

Session altered.

SQL> SELECT SYSTIMESTAMP FROM DUAL;

SYSTIMESTAMP
_____________________________________
27-JAN-22 02.14.47.578946000 PM -05:00
```

> **Note:**
>
> When a `SYSDATE` or `SYSTIMESTAMP` query is executed in SQL Worksheet of Database Actions, the time and date value that is returned is in UTC (when `SYSDATE_AT_DBTIMEZONE` parameter is set to `TRUE` or `FALSE`). To obtain the database time zone when working in Database Actions, use `TO_CHAR()` as follows:
>
> ```
> SQL> SELECT TO_CHAR(SYSTIMESTAMP,'YYYY-MM-DD"T"HH24:MI:SS TZH":"TZM')
> FROM DUAL;
>
> TO_CHAR(SYSTIMESTAMP,'YYYY-MM-DD"T"HH24:MI:SSTZH":"TZM')
> ____________________________________________________________
> 2022-01-27T14:15:00 -05:00
> ```

CLIENT_PREFETCH_ROWS

Set the `CLIENT_PREFETCH_ROWS` parameter to enable clients to reduce the number of roundtrips required while fetching rows of a query result set.

`CLIENT_PREFETCH_ROWS` specifies the number of rows to be prefetched by the Oracle client driver, without making any changes to the client application. The client driver buffers the prefetched rows after each successful query execution and for each subsequent fetch request sent to the database.

This parameter applies only to clients that use Oracle Call Interface (OCI) to connect to the database.

This parameter applies only with Oracle Instant Client/Oracle Database Client 19.17 (or later) and 21.8 (or later), for all platforms.

Property	Description
Parameter type	Integer
Syntax	`CLIENT_PREFETCH_ROWS = integer`
Default Value	`0` (Only client-side settings apply)
Modifiable	`ALTER SYSTEM`, `ALTER SESSION`
Range of values	`0` to `UB4MAXVAL` (`4294967295`)
Basic	No

The `CLIENT_PREFETCH_ROWS` parameter can be set with `ALTER SESSION` or `ALTER SYSTEM`. If the parameter value changes using `ALTER SESSION`, the new value becomes effective for that specific session on subsequent resultset fetches. If the parameter value changes using `ALTER SYSTEM`, the new value takes effect for the statements that run on connections created after the `ALTER SYSTEM` command.

For example, if `CLIENT_PREFETCH_ROWS` is set to 100 and a client application asks to fetch 10 rows, a total of 110 rows are returned to the client driver. The first 10 rows out of the 110 rows are given to the application, and the client driver internally buffers the remaining 100 rows. The next 10 row fetches from the client application, each with 10 rows per fetch iteration can be fulfilled from the 100 rows that are internally buffered by the client driver. This process reduces the number of required network roundtrips to and from the database. In this example, on the

11th fetch, a new network roundtrip is incurred and the database returns the next batch of 110 rows, as long as the result set is not exhausted, and the cycle repeats.

Notes for setting `CLIENT_PREFETCH_ROWS`:

- When `CLIENT_PREFETCH_ROWS` is set to a non-zero value, its value takes precedence over the default `OCI_ATTR_PREFETCH_ROWS` value for prefetch row count.
- If the `OCI_ATTR_PREFETCH_ROWS` value is set to a non-default value, then the `CLIENT_PREFETCH_ROWS` value is ignored for the prefetch row count.
- Using `CLIENT_PREFETCH_ROWS` with `OCIAttrSet()`:

  `OCI_ATTR_PREFETCH_ROWS` sets the number of top-level rows to be prefetched. The default value is 1 row. However, if `CLIENT_PREFETCH_ROWS` is set, the number of top-level rows to be prefetched is determined by the following precedence:

  1. If you set the `OCI_ATTR_PREFETCH_ROWS` attribute using `OCIAttrSet()` function or `oraaccess.xml` as the value '1', then the database initialization parameter `CLIENT_PREFETCH_ROWS` value takes precedence and determines the number of top-level rows to be prefetched.
  2. If you set the `OCI_ATTR_PREFETCH_ROWS` attribute using `OCIAttrSet()` function or `oraaccess.xml` as the value '*x*' other than 1, then '*x*' number of top-level rows will be prefetched, and the database initialization parameter `CLIENT_PREFETCH_ROWS` is ignored.
  3. If you do not set an `OCI_ATTR_PREFETCH_ROWS` value using `OCIAttrSet()` or `oraaccess.xml`, then the database initialization parameter `CLIENT_PREFETCH_ROWS` value takes precedence and determines the number of top-level rows to be prefetched.
- Using `CLIENT_PREFETCH_ROWS` with `OCIAttrGet()`:

  The function `OCIAttrGet()` returns the effective prefetch row value set from `OCI_ATTR_PREFETCH_ROWS`, `oraaccess.xml` and the database initialization parameter `CLIENT_PREFETCH_ROWS`. If the `OCI_ATTR_PREFETCH_MEMORY` value is set, the value returned by `OCIAttrGet()` might not be the final prefetch rows value and may be restricted to the maximum number of rows allowed by the memory value specified by the `OCI_ATTR_PREFETCH_MEMORY` attribute.

See Also:

- *Oracle Call Interface Developer's Guide* for more information about fetching results and setting the prefetch count.
- OCIAttrSet()
- OCI_ATTR_PREFETCH_ROWS

JOB_QUEUE_PROCESSES

Set the `JOB_QUEUE_PROCESSES` parameter to specify the maximum number of job workers that can be created to run Oracle Scheduler (`DBMS_SCHEDULER`) jobs.

Setting the value to 0 disables non-Oracle supplied Scheduler jobs.

Property	Description
Parameter type	Integer
Syntax	`JOB_QUEUE_PROCESSES = integer`
Default Value	The default value depends on the **ECPU count** (**OCPU count** if your database uses OCPUs) and the setting for **ECPU auto scaling** (**OCPU auto scaling** if your database uses OCPUs). See the following table for details.
Modifiable	`ALTER SYSTEM`
Range of values	Minimum value: `0` Maximum value depends on the **ECPU count** (**OCPU count** if your database uses OCPUs) and the setting for **ECPU auto scaling** (**OCPU auto scaling** if your database uses OCPUs), as shown in the following table.

The default and maximum values for `JOB_QUEUE_PROCESSES` differ depending on the compute model you use. See Compute Models in Autonomous Database for more information.

Compute Model	Default and Maximum Value with Auto Scaling Disabled	Default and Maximum Value with Auto Scaling Enabled
ECPU	7.5 x ECPU count	22.5 x ECPU count
OCPU	30 x OCPU count	90 x OCPU count

Oracle Scheduler job coordinator and job workers are controlled by the `JOB_QUEUE_PROCESSES` parameter. The actual number of job workers created for Oracle Scheduler jobs is auto-tuned by the Scheduler depending on several factors, including available resources, Resource Manager settings, and currently running jobs.

The default value for `JOB_QUEUE_PROCESSES` provides a compromise between quality of service for applications and reasonable use of system resources. However, it is possible that the default value does not suit every environment.

Setting the value of `JOB_QUEUE_PROCESSES` to 0 disables non-Oracle supplied Scheduler jobs. When `JOB_QUEUE_PROCESSES` is set to `0` this does not disable any internal jobs for Oracle-supplied users (service related jobs run by Oracle-supplied users continue to be scheduled).

Oracle-supplied users are users marked as `ORACLE_MAINTAINED` with value `Y`. Non Oracle-supplied users are users marked as `ORACLE_MAINTAINED` with value `N`.

See ALL_USERS for more information.

RESULT_CACHE_INTEGRITY

Set the `RESULT_CACHE_INTEGRITY` parameter to specify whether the result cache considers queries using possibly non-deterministic constructs as candidates for result caching.

Property	Description
Parameter type	String
Syntax	`RESULT_CACHE_INTEGRITY = { ENFORCED | TRUSTED }`
Default Value	For Autonomous Database the default value is: `ENFORCED`
Modifiable	`ALTER SYSTEM`

Values

- `ENFORCED`: Irrespective of the setting of `RESULT_CACHE_MODE` or specified hints, only deterministic constructs are eligible for result caching. For example, queries using PL/SQL functions that are not declared as deterministic are never cached (unless the functions are declared as deterministic the query results with such functions will not be cached).
- `TRUSTED`: The database honors the setting of `RESULT_CACHE_MODE` and specified hints and will consider queries using possibly non-deterministic constructs as candidates for result caching. For example, queries using PL/SQL functions that are not declared as deterministic can be cached. Note, however, that results that are known to be non-deterministic are not cached (for example `SYSDATE` or constructs involving `SYSDATE`).

RESULT_CACHE_MODE

Set the `RESULT_CACHE_MODE` parameter to specify which queries are eligible to store result sets in the result cache. Only query execution plans with the result cache operator will attempt to read from or write to the result cache.

Property	Description
Parameter type	String
Syntax	`RESULT_CACHE_MODE = { MANUAL | MANUAL_TEMP | FORCE | FORCE_TEMP }`
Default Value	For Autonomous Database with workload type set to Data Warehouse: `FORCE` For workload types Transaction Processing, JSON, or APEX: `MANUAL`
Modifiable	`ALTER SESSION`, `ALTER SYSTEM`

See RESULT_CACHE_MODE for more information.

- Using SQL Result Cache Hints
  Use result cache hints at the application level to control caching behavior. The SQL result cache hints take precedence over the result cache mode and result cache table annotations.

Using SQL Result Cache Hints

Use result cache hints at the application level to control caching behavior. The SQL result cache hints take precedence over the result cache mode and result cache table annotations.

You can use SQL result cache hints in the following ways:

- Using the `RESULT_CACHE` Hint
- Using the `NO_RESULT_CACHE` Hint
- Using the `RESULT_CACHE` Hint in Views

Using the `RESULT_CACHE` Hint

When the result cache mode is `MANUAL`, the `/*+ RESULT_CACHE */` hint instructs the database to cache the results of a query block and to use the cached results in future executions.

See Using the RESULT_CACHE Hint for more information.

Using the `NO_RESULT_CACHE` Hint

The `/*+ NO_RESULT_CACHE */` hint instructs the database not to cache the results in either the server or client result caches.

See Using the NO_RESULT_CACHE Hint for more information.

Using the `RESULT_CACHE` Hint in Views

The `RESULT_CACHE` hint applies only to the query block in which the hint is specified. If the hint is specified only in a view, then only these results are cached.

See Using the RESULT_CACHE Hint in Views for more information.

SQL Commands

Autonomous Database allows most of the SQL commands available in Oracle Database. To ensure the security and the performance of Autonomous Database, some SQL commands are restricted.

This section provides a list of SQL command limitations that are required to protect security and for the performance integrity of Autonomous Databases. Most of the standard SQL and PL/SQL syntax and constructs available with Oracle Database work in Autonomous Databases.

> **Note:**
>
> If you try to use a restricted SQL command the system reports:
>
> ```
> ORA-01031: insufficient privileges
> ```
>
> This error indicates that you are not allowed to run the SQL command in Autonomous Database.

The following SQL statements are not available in Autonomous Database:

- `ADMINISTER KEY MANAGEMENT`: By default Autonomous Database uses Oracle-managed encryption keys. Using Oracle-managed keys, Autonomous Database creates and manages the encryption keys that protect your data and Oracle handles rotation of the TDE master key.

  If you want customer-managed keys, a master encryption key in the Oracle Cloud Infrastructure Vault is used to generate the TDE master key on Autonomous Database. See Managing Encryption Keys on Autonomous Database for more information.
- `CREATE TABLESPACE`, `ALTER TABLESPACE`, and `DROP TABLESPACE`: Autonomous Database automatically configures default data and temporary tablespaces for the database. Adding, removing, or modifying tablespaces is not allowed. Autonomous Database creates one tablespace or multiple tablespaces automatically depending on the storage size.
- `CREATE DATABASE LINK`

  Use `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to create database links in Autonomous Database. See Use Database Links with Autonomous Database for more information.

- CREATE LIBRARY

SQL Statements with Restrictions in Autonomous Database

The following DDL statements are available in Autonomous Database with some restrictions:

SQL Command	Restrictions
ALTER PLUGGABLE DATABASE and ALTER DATABASE	Only the following clauses are allowed: DATAFILE AUTOEXTEND ON DATAFILE AUTOEXTEND OFF DEFAULT EDITION SET TIME_ZONE SET CMU_WALLET
ALTER PROFILE	Using ALTER PROFILE, there are restrictions for a user defined PASSWORD_VERIFY_FUNCTION. See Manage Password Complexity on Autonomous Database for more information. Using ALTER PROFILE, the optional CONTAINER clause is ignored if specified. See Create Users on Autonomous Database for information on the password parameter values defined in the default profile.
ALTER SESSION	Only the following clauses are allowed: ADVISE COMMIT, ADVISE ROLLBACK, ADVISE NOTHING CLOSE DATABASE LINK ENABLE COMMIT IN PROCEDURE, DISABLE COMMIT IN PROCEDURE ENABLE PARALLEL <QUERY|DDL|DML>, DISABLE PARALLEL <QUERY|DDL|DML>, FORCE PARALLEL <QUERY|DDL|DML> ENABLE RESUMABLE, DISABLE RESUMABLE SET CONSTRAINTS SET CURRENT_SCHEMA SET DEFAULT_COLLATION SET EDITION SET ISOLATION_LEVEL SET OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES SET ROW ARCHIVAL VISIBILITY SET STATISTICS_LEVEL SET TIME_ZONE
ALTER SYSTEM	ALTER SYSTEM is not allowed except ALTER SYSTEM SET and ALTER SYSTEM KILL SESSION SET can only be used to set parameters listed in Initialization Parameters.
ALTER USER	The following clause is ignored: DEFAULT TABLESPACE The IDENTIFIED with the EXTERNALLY clause is not supported. The IDENTIFIED BY VALUES clause is not allowed.
ALTER TABLE	For restrictions, see ALTER TABLE Restrictions.

SQL Command	Restrictions
CREATE PROFILE	PASSWORD_VERIFY_FUNCTION See Manage Password Complexity on Autonomous Database for more information. Using ALTER PROFILE, the optional CONTAINER clause is ignored if specified. See Create Users on Autonomous Database for information on the password parameter values defined in the default profile.
CREATE TABLE	For restrictions, see CREATE TABLE Restrictions.
CREATE USER	The following clause is ignored: • DEFAULT TABLESPACE IDENTIFIED with the EXTERNALLY clause is not supported. The IDENTIFIED BY VALUES clause is not allowed.

CREATE TABLE Restrictions

XMLType tables using XML schema-based storage are not allowed. See Oracle XML DB for more information.

The clauses not in this list are allowed.

Clause	Comment
cluster	Ignored
ilm_clause	Ignored
inmemory_table_clause	Ignored
LOB_storage_clause	The LOB_compression_clause is recognized. Other LOB_storage_clause parameters are ignored. See LOB_compression_clause for more information.
logging_clause	Ignored
organization external	Ignored
organization index	Creates a regular table with a primary key. Using the organization index clause does not create an index-organized table. You should test and verify the performance of the generated table for your application.
physical_properties	Ignored

> **Note:**
>
> For more information on CREATE TABLE, see *Database SQL Language Reference*.

ALTER TABLE Restrictions

The clauses not in this list are allowed.

Clause	Comment
allocate_extent_clause	Ignored

Clause	Comment
alter_iot_clauses	Ignored
deallocate_unused_clause	Ignored
ilm_clause	Ignored
inmemory_table_clause	Ignored
logging_clause	Ignored
modify_LOB_storage_clause	Ignored
physical_attributes_clause	Ignored
shrink_clause	Ignored

> **Note:**
>
> For more information on ALTER TABLE, see *Database SQL Language Reference*.

Data Types

Autonomous Database allows most of the data types available in Oracle Database. To ensure the security and the performance of Autonomous Database, some data types are restricted.

The following data types are not supported or have limited support in Autonomous Database:

- Large Object (LOB) data types: only SecureFiles LOB storage is supported. BasicFiles LOBs are automatically converted to SecureFiles LOBs.
- Media types are not supported (Oracle Multimedia is desupported)

Checking and Setting MAX_STRING_SIZE

By default Autonomous Database uses extended data types and the value of `MAX_STRING_SIZE` is set to the value `EXTENDED`. With this setting you can specify a maximum size of 32767 bytes for the `VARCHAR2`, `NVARCHAR2`, and `RAW` data types. The default, `EXTENDED`, is the recommended setting and allows Autonomous Database to take full advantage of database capabilities.

Use `DBMS_MAX_STRING_SIZE` subprograms to check usage of extended data types and to change the database to revert to the older style `STANDARD`, supporting a maximum size of 4000 bytes for `VARCHAR2`, `NVARCHAR2`, and `RAW` data types.

> **Note:**
>
> Using `DBMS_MAX_STRING_SIZE.MODIFY_MAX_STRING_SIZE` is a one-way change that cannot be reverted. After a database is switched back to the `STANDARD` style of supporting a maximum length of 4000 bytes for the `VARCHAR2`, `NVARCHAR2`, and `RAW` data types, you cannot re-enable `EXTENDED` data types.
>
> The ADMIN user is granted `EXECUTE` privilege `WITH GRANT OPTION` clause on `DBMS_MAX_STRING_SIZE`. Oracle recommends that you do not `GRANT EXECUTE` on this package to other users.

1. Check whether your environment can be reverted to the old style, `STANDARD` behavior:

   ```
   SELECT * FROM
   TABLE(DBMS_MAX_STRING_SIZE.CHECK_MAX_STRING_SIZE('STANDARD'));
   ```

   See CHECK_MAX_STRING_SIZE Function for more information.
2. Check and correct all reported violations from Step 1, if applicable.
3. After fixing any reported violations found in Step 1, if you want to revert to a maximum length of 4000 bytes for `VARCHAR2`, `NVARCHAR2`, and `RAW` data types, use `DBMS_MAX_STRING_SIZE.MODIFY_MAX_STRING_SIZE` as follows:

   ```
   EXEC DBMS_MAX_STRING_SIZE.MODIFY_MAX_STRING_SIZE('STANDARD');
   ```

   See MODIFY_MAX_STRING_SIZE Procedure for more information.

See Extended Data Types for details on extended data types.

For a list of Oracle data types see *Oracle Database SQL Language Reference*.

PL/SQL Packages Notes for Autonomous Database

Notes for Oracle Database PL/SQL packages in Autonomous Database.

Unavailable PL/SQL Packages

- `DBMS_DEBUG_JDWP`
- `DBMS_DEBUG_JDWP_CUSTOM`
- `UTL_INADDR`

PL/SQL Packages Notes

- **`DBMS_LDAP`**
  - Specifying an IP address in the host name is not allowed.
  - The only allowed port is 636.
  - The `SSLWRL` and `SSLWALLETPASSWD` arguments to the `OPEN_SSL` procedure are ignored. The default value for the `SSLWRL` property is set to the wallet that is used by `UTL_HTTP` and `DBMS_CLOUD` for making outbound web requests on Autonomous Database.
  - The `DBMS_LDAP.SIMPLE_BIND_S` and `DBMS_LDAP.BIND_S` subprograms perform authentication to the directory server.

    The `DBMS_LDAP.SIMPLE_BIND_S` and `DBMS_LDAP.BIND_S` subprograms are modified to accept credential objects as an argument.

    Following are the usage notes and examples of these modified subprograms:

    * The modified `SIMPLE_BIND_S` and `BIND_S` subprograms enable you to pass credential objects to set directory server authentication. Credential objects are schema objects, hence they can be accessed only by privileged users and enable you to configure schema-level privileges to access control the credentials. Passing scheduler credentials is an appropriate and secure way to store and manage username/password/keys for authentication.

* The modified `SIMPLE_BIND_S` and `BIND_S` subprograms are a secure and convenient alternative to previously existed `SIMPLE_BIND_S` and `BIND_S` subprogram.

  See FUNCTION simple_bind_s and FUNCTION bind_s for more information.

* The `CREDENTIAL` argument of the `SIMPLE_BIND_S` and `BIND_S` functions is used to perform credential based authentication to the directory server.

* For example:

  * Create a credential object:

    ```
    BEGIN DBMS_CLOUD.CREATE_CREDENTIAL (
        credential_name => 'LDAP_CRED',
        username        => 'web_app_user',
        password        => 'password' );
    END;
    ```

    This creates a credential object which creates a stored username/password pair.

    See CREATE_CREDENTIAL Procedure for more information.

    See Specifying Scheduler Job Credentials for more information.

  * Invoke `DBMS_LDAP.SIMPLE_BIND_S`:

    ```
    DECLARE
          l_mail_conn DBMS_LDAP.INIT;
        BEGIN
          l_ldap_conn   := DBMS_LDAP.INIT('ldap.example.com', 636);
          l_auth_result := DBMS_LDAP.SIMPLE_BIND_S(l_ldap_conn,
    'LDAP_CRED');
          ...
    END;
    ```

    The code in this example first invokes the `DBMS_LDAP.INIT` function which initializes a session with an LDAP server and establishes a connection with the LDAP server `ldap.example.com` at port number `636`. The value `l_ldap_conn` in the `SIMPLE_BIND_S` function is the LDAP session handle and `LDAP_CRED` is the credentials name.

  * The `function bind_s` performs complex authentication to the directory server. For example:

    ```
    DECLARE
          l_mail_conn DBMS_LDAP.INIT;
        BEGIN
          l_ldap_conn   := DBMS_LDAP.INIT('ldap.example.com', 636);
          l_auth_result := DBMS_LDAP.BIND_S(l_ldap_conn,
    'LDAP_CRED', METH => DBMS_LDAP.AUTH_SIMPLE);
          ...
    END;
    ```

    The code in this example first invokes the `DBMS_LDAP.INIT` function which initializes a session with an LDAP server and establishes a connection with the

LDAP server `ldap.example.com` at port number `636`. The value `l_ldap_conn` in the `BIND_S` function is the LDAP session handle and `LDAP_CRED` is the credentials name. `METH` is the authentication method. The only valid value is `DBMS_LDAP_UTL.AUTH_SIMPLE`.

  - The `EXECUTE` privileges on `DBMS_CLOUD` or `DWROLE` is required to create scheduler credentials.
  - The passed credentials must be present in the current user schema and be in the enabled state.
  - A public or private synonym that points to a credential in a different user schema can be supplied as a value for the `CREDENTIAL` parameter provided you have the `EXECUTE` privilege on the base credential object pointed to by the synonym. See Overview of Synonyms for more information.

- `DBMS_LDAP` usage is audited by default. You cannot disable auditing for `DBMS_LDAP`.
- SSL/TLS is enforced for all communication happening between LDAP server and Autonomous Database.
- When your Autonomous Database instance is configured with a private endpoint, set the `ROUTE_OUTBOUND_CONNECTIONS` database parameter to '`PRIVATE_ENDPOINT`' to specify that all outgoing LDAP connections are subject to the Autonomous Database instance private endpoint VCN's egress rules. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

To use `DBMS_LDAP` for a connection on a private endpoint, use `DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE` and specify the `private_target` parameter with value `TRUE`.

> **Note:**
>
> If you set `ROUTE_OUTBOUND_CONNECTIONS` to `PRIVATE_ENDPOINT`, setting the `private_target` parameter to `TRUE` is not required in this API. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

- **`UTL_HTTP`**
  - Connections through IP addresses are not allowed.
  - Only `HTTPS` is allowed when the Autonomous Database instance is on a public endpoint. When the Autonomous Database instance is on a private endpoint, both `HTTPS` and `HTTP_PROXY` connections are allowed (`HTTP` connections are disallowed for both public endpoints and private endpoints).
  - The `UTL_HTTP.set_proxy` API is allowed when the Autonomous Database instance is on a private endpoint.
  - When the Autonomous Database instance is on a private endpoint and you use `HTTP_PROXY` or the `UTL_HTTP.SET_PROXY` API:
    - `DBMS_CLOUD` requests do not honor the proxy server you set with `UTL_HTTP.SET_PROXY`. This includes `DBMS_CLOUD.SEND_REQUEST` and all object storage access for `DBMS_CLOUD` external tables that you define with `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`, `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`, or `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE`.
    - `APEX_WEB_REQUEST` requests do not honor the proxy server you set with `UTL_HTTP.SET_PROXY`.

- All web services must be secured. The only allowed port is 443 when the Autonomous Database instance is on a public endpoint. When the Autonomous Database instance is on a private endpoint this restriction does not apply.

  Your Autonomous Database instance is preconfigured with an Oracle Wallet that contains more than 90 of the most commonly trusted root and intermediate SSL certificates. The Oracle Wallet is centrally managed.
- The `SET_AUTHENTICATION_FROM_WALLET` procedure is disallowed.
- The `WALLET_PATH` and `WALLET_PASSWORD` arguments for the `CREATE_REQUEST_CONTEXT`, `REQUEST`, and `REQUEST_PIECES` procedures are ignored.
- The `CREDENTIAL` argument of the `SET_CREDENTIAL` procedure is used to pass the credential object as an input to the procedure. See Specifying Scheduler Job Credentials and CREATE_CREDENTIAL Procedure for more information.
- The `EXECUTE` privileges on `DBMS_CLOUD` or `DWROLE` is required to create credential objects.
- The passed credentials must be present in the current user schema and be in the enabled state.
- A public or private synonym that points to a credential in a different user schema can be supplied as a value for the `CREDENTIAL` parameter provided you have the `EXECUTE` privilege on the base credential object pointed to by the synonym. See Overview of Synonyms for more information.
- Oracle Wallet configuration cannot be altered. All arguments for `SET_WALLET` procedure are ignored.
- `UTL_HTTP` usage is audited by default. You cannot disable auditing for `UTL_HTTP`.
- When your Autonomous Database instance is configured with a private endpoint, set the `ROUTE_OUTBOUND_CONNECTIONS` database parameter to '`PRIVATE_ENDPOINT`' to specify that all outgoing `UTL_HTTP` connections are subject to the Autonomous Database instance private endpoint VCN's egress rules. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

- **`UTL_SMTP`**
  - The only supported email provider is Oracle Cloud Infrastructure Email Delivery service. See Overview of the Email Delivery Service for more information.
  - Mail with an IP address in the host name is not allowed.
  - The only allowed ports are 25 and 587.
  - The `CREDENTIAL` argument of the `SET_CREDENTIAL` function is used to pass the scheduler credentials object as an input to the function. See Specifying Scheduler Job Credentials and CREATE_CREDENTIAL Procedure for more information.
  - The `EXECUTE` privileges on `DBMS_CLOUD` or `DWROLE` is required to create credential objects.
  - The `CREDENTIAL` argument of the `SET_CREDENTIAL` procedure is used to pass the credential objects object as an input to the procedure. See Specifying Scheduler Job Credentials for more information.
  - The passed credentials must be present in the current user schema and be in the enabled state.
  - A public or private synonym that points to a credential in a different user schema can be supplied as a value for the `CREDENTIAL` parameter provided you have the `EXECUTE`

privilege on the base credential object pointed to by the synonym. See Overview of Synonyms for more information.

  - `UTL_SMTP` usage is audited by default. You cannot disable auditing for `UTL_SMTP`.
  - When your Autonomous Database instance is configured with a private endpoint, set the `ROUTE_OUTBOUND_CONNECTIONS` database parameter to '`PRIVATE_ENDPOINT`' to specify that all outgoing `UTL_SMTP` connections are subject to the Autonomous Database instance private endpoint VCN's egress rules. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

- **`UTL_TCP`**
  - The IP address is not allowed in the host name.
  - The only allowed ports are: 443 (HTTP) 25 and 587 (SMTP).
  - For port 443, only HTTPS URLs are allowed.
  - The `WALLET_PATH` and `WALLET_PASSWORD` arguments for the `OPEN_CONNECTION` procedure are ignored. The default value for the `WALLET_PATH` and `WALLET_PASSWORD` property are set to the wallet that is used by `UTL_HTTP` and `DBMS_CLOUD` for making outbound web requests on Autonomous Database.
  - `UTL_TCP` usage is audited by default. You cannot disable auditing for `UTL_TCP`.
  - SSL/TLS is enforced for all communication happening over TCP/IP connections.
  - When your Autonomous Database instance is configured with a private endpoint, set the `ROUTE_OUTBOUND_CONNECTIONS` database parameter to '`PRIVATE_ENDPOINT`' to specify that all outgoing `UTL_TCP` connections are subject to the Autonomous Database instance private endpoint VCN's egress rules. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

- **`DBMS_NETWORK_ACL_ADMIN`**
  - Granting ACL privileges on IP addresses is not allowed.
  - The `HTTP_PROXY` ACL is allowed on private endpoints.

- **`UTL_HTTP` Errors**

  The following table shows error messages and possible causes for these error messages when using `UTL_HTTP`:

Error Message	Potential Cause
`ORA-12545: Connect failed because target host or object does not exist`	Target host or object does not exist or it is private.
`ORA-24247: network access denied by access control list (ACL)`	Access control list (ACL) for the specified host could not be found.
`ORA-29024: Certificate validation failure`	Certificate of the host does not exist or is not among the supported certificates.
`ORA-29261: Bad argument`	Passed credentials are invalid or disabled or the user does not have sufficient privileges on the credential.

- **`UTL_SMTP` Error**

Error Message	Potential Cause
`ORA-29261: Bad argument`	Passed credentials are invalid or disabled or the user does not have sufficient privileges on the credential.

- **`DBMS_LDAP` Error**

Error Message	Potential Cause
`ORA-31400: Missing or invalid scheduler credential`	Passed credentials are NULL or invalid.

See UTL_HTTP, DBMS_LDAP, UTL_SMTP, UTL_TCP, and DBMS_NETWORK_ACL_ADMIN in *PL/SQL Packages and Types Reference* for more information.

Oracle XML DB

Describes Autonomous Database support for Oracle XML DB features. To ensure the security and the performance of your Autonomous Database, some Oracle XML DB features are restricted.

The following is supported, in addition to the features listed:

- Full support for XMLQuery, XMLTable, and other SQL/XML standard functions
- Indexing schema including functional indexes using SQL/XML expressions, Structured XMLIndex and XQuery Full Text Index

> **Note:**
>
> If you migrate tables containing `XMLType` columns to Autonomous Database using Oracle Data Pump, you need to convert to Non-Schema Binary XML prior to using Oracle Data Pump Export (expdp).

Area	XML DB Feature	Supported in Autonomous Database	More Information
Repository	XML DB Protocol	No	Repository Access Using Protocols
Repository	XML DB Resources	No	Oracle XML DB Repository Resources
Repository	XML DB ACLs	No	Repository Access Control
Storage	XML Schema Registration	No	XML Schema Registration with Oracle XML DB
Storage	CLOB	No	Deprecated
Storage	Object Relational	No	XML Schema and Object-Relational XMLType
Storage	Binary XML	Yes (Non schema-based only)	XMLType Storage Models
Index	Structured XML Index	Yes	XMLIndex Structured Component
Index	XQuery Full Text Index	Yes	Indexing XML Data for Full-Text Queries
Index	Unstructured XMLIndex	No	XMLIndex Unstructured Component

Area	XML DB Feature	Supported in Autonomous Database	More Information
Packages	XML DOM package	Yes	PL/SQL DOM API for XMLType (DBMS_XMLDOM)
Packages	XML Parser Package	Yes	PL/SQL Parser API for XMLType (DBMS_XMLPARSER)
Packages	XSL Processor (DBMS_XSLPROCESSOR)	Yes	PL/SQL XSLT Processor for XMLType (DBMS_XSLPROCESSOR)

For details on Oracle XML DB, see Oracle XML DB Developer's Guide.

Oracle Text

Describes Autonomous Database support for Oracle Text features. To ensure the security and the performance of your Autonomous Database, some Oracle Text features are restricted.

Oracle Text Feature	Supported in Autonomous Database	More Information
All logging, and APIs which perform logging such as `ctx_report.query_log_summary`	Not Supported	QUERY_LOG_SUMMARY
File Datastore	Not Supported (see replacement `DIRECTORY_DATASTORE`)	Datastore Types
URL Datastore	Not Supported (see replacement `NETWORK_DATASTORE`)	Datastore Types
`CREATE INDEX` with `BIG_IO` option	Supported if you grant the privilege to create a trigger to the user (`GRANT CREATE TRIGGER`).	Improved Response Time Using the BIG_IO Option of CONTEXT Index
`OPTIMIZE_INDEX` in rebuild mode	Supported if you grant the privilege to create a trigger to the user (`GRANT CREATE TRIGGER`).	OPTIMIZE_INDEX

For details on Oracle Text, see Oracle Text Application Developer's Guide.

Oracle Flashback

Oracle Flashback Technology is a group of Oracle Database features that let you view past states of database objects or to return database objects to a previous state without using point-in-time media recovery.

To restore and recover your database to a point in time, see Restore and Recover your Autonomous Database.

Oracle Flashback Feature	Supported in Autonomous Database
DBMS_FLASHBACK	Yes except the procedure: DBMS_FLASHBACK.TRANSACTION_BACKOUT
Flashback Data Archive	Yes
Flashback Drop	Yes
Flashback Query	Yes
Flashback Table	Yes
Flashback Transaction	No
Flashback Transaction Query	Yes
Flashback Version Query	Yes

See Track Table Changes with Flashback Time Travel for information on using Flashback features.

Oracle Database Real Application Security

Oracle Database Real Application Security is a database authorization model that: supports declarative security policies, enables end-to-end security for multitier applications, provides an integrated solution to secure database and application resources, and advances the security architecture of Oracle Database to meet existing and emerging demands of applications developed for the Internet.

See Introducing Oracle Database Real Application Security more information.

Real Application Security works the same on Autonomous Database as on an on-premises Oracle Database except you need to perform the following ADMIN tasks before using Real Application Security on Autonomous Database:

- To create Real Application Security users/roles, you need the PROVISION system privilege. As the ADMIN user run the following command to grant this privilege to a database user:

  ```
  SQL> EXEC
  XS_ADMIN_CLOUD_UTIL.GRANT_SYSTEM_PRIVILEGE('PROVISION','DB_USER');
  ```

  In this example, *DB_USER* is a database user.

  Running this command on Autonomous Database replaces the following on-premise database command (note the _CLOUD_ is not in the following package name):

  ```
  SQL> EXEC SYS.XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE('PROVISION', 'DB_USER',
  XS_ADMIN_UTIL.PTYPE_XS);
  ```

  See General Procedures for Creating Application User Accounts for more information.

- To create Real Application Security data controls, you need the ADMIN_ANY_SEC_POLICY privilege. As the ADMIN user run the following command to grant this privilege:

  ```
  EXEC
  XS_ADMIN_CLOUD_UTIL.GRANT_SYSTEM_PRIVILEGE('ADMIN_ANY_SEC_POLICY','DB_USER'
  );
  ```

In this example, *DB_USER* is a database user.

Running this command on Autonomous Database replaces the following on-premise database command (note the _CLOUD_ is not in the following package name):

```
SQL> EXEC
SYS.XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE('ADMIN_ANY_SEC_POLICY','DB_USER');
```

See Creating Roles and Application Users for more information.

Oracle LogMiner

Describes restrictions for Oracle LogMiner on Autonomous Database.

Archived Log Retention Maximum is 48 Hours

Autonomous Database archived log files are kept for up to 48 hours. LogMiner can only access up to 48 hours of archived log files.

If you attempt to mine log files older than 48 hours, LogMiner reports `ORA-1285: "error reading file"`.

Choose a Character Set for Autonomous Database

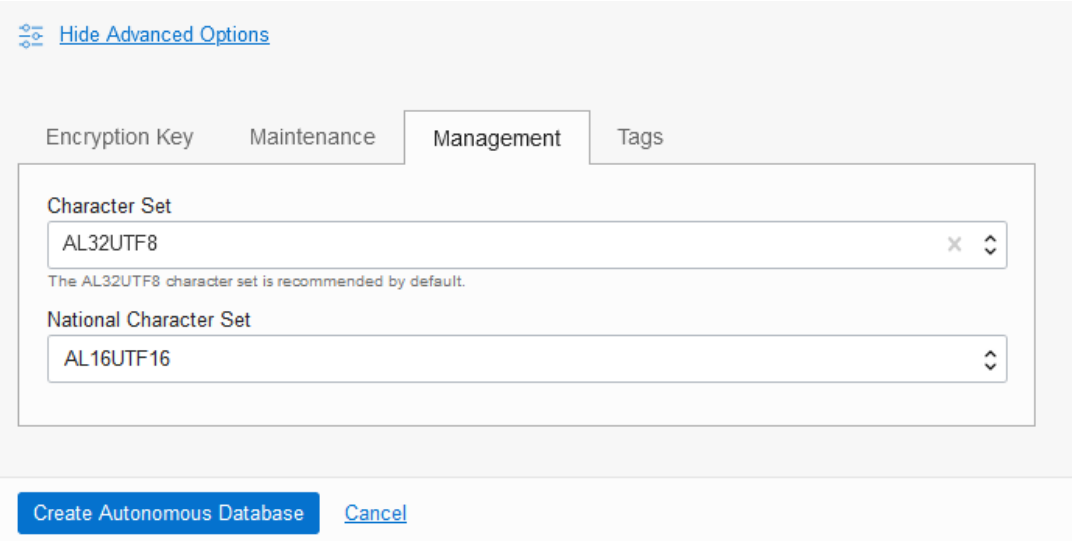
The Autonomous Database default database character set is Unicode AL32UTF8 and the default national character set is AL16UTF16. When you provision a database, depending on the workload type, you can select a database character set and a national character set.

> **Note:**
>
> Oracle recommends using the default Unicode database character set (AL32UTF8) for its universality and compatibility with contemporary and future technologies and language requirements.

If you are using an on-premises database with a non-Unicode character set, migrating to the default Unicode character set can be a convoluted process requiring complex data analysis. Thus, Autonomous Database lets you choose a character set when you provision an Autonomous Database instance.

1. Create an Autonomous Database following the provisioning steps.

   See Provision Autonomous Database for more information.
2. On the **Create Autonomous Database** page expand **Show Advanced Options**.
3. In the advanced options area, select the **Management** tab.



- In the **Character Set** field, use the selector to choose a character set.
- In the **National Character Set** field, use the selector to choose a national character set.

To make selecting a value easier, when you type in the text area this filters the list. For example, if you type JA you see only the options containing JA, including: JA16EUC, JA16EUCTILDE, JA16SJIS, JA16JA16SJISTILDE, and JA16VMS.

4. Click **Create Autonomous Database** to provision the Autonomous Database.

See the following for more information:

- Choosing an Oracle Database Character Set
- Character Set Migration
- Support Note 788156.1
- Notes for Character Set Selection
  Provides notes and limitations for selecting a character set and a national character set on Autonomous Database.

Notes for Character Set Selection

Provides notes and limitations for selecting a character set and a national character set on Autonomous Database.

- When you provision an Autonomous Database instance you can only select a character set for **Data Warehouse**, **Transaction Processing**, or **APEX** workload types.

  The **JSON Database** workload type only supports the default character set and you cannot select a different character set.
- Always Free Autonomous Database does not support character set selection.
- You cannot select a different character set when you clone an instance. A cloned Autonomous Database instance has the same character set as the source database.
- You cannot change the character set of an existing Autonomous Database instance.
- APEX developer and administration pages are supported in English only when the database character set is different from the default (AL32UTF8).

Translation of user applications into other languages or setting an application's primary language to a value other than English is not supported. If you use APEX in a language other than English, you may experience issues such as illegible, garbage text. You can, however, use your APEX applications to process non-English user data, as long as the languages of the data are supported by the selected database character set.

If you need full globalization support in APEX, migrate your applications and data to AL32UTF8, the universal Unicode character set, which is the default and recommended character set for Autonomous Database.

When you choose a character set other than the default Unicode character set (AL32UTF8) on Autonomous Database, the APEX language selector only shows languages that are supported in that database character set. For example, if you choose the database character set WE8ISO8859P1 (ISO 8859-1 West European), the language selector does not show Japanese or Korean.

- While you are provisioning an instance the **Character Set** selector on the **Management** tab lists the supported character set names. If you want to see a list of supported database character sets before you provision a database, refer to the following:
  - See Table A-4 in Recommended Database Character Sets
  - See Table A-6 in Other Character Sets

Database Features Unavailable in Autonomous Database

Lists the Oracle Database features that are not available in Autonomous Database. Additionally, database features designed for administration are not available.

List of Unavailable Oracle Features

- Oracle Real Application Security Administration Console (RASADM)
- Oracle Industry Data Models
- Oracle Database Lifecycle Management Pack
- Oracle Data Masking and Subsetting Pack

> **Note:**
>
> Oracle Data Safe, available with Autonomous Database provides Data Masking. See Use Oracle Data Safe with Autonomous Database for more information.

- Oracle Cloud Management Pack for Oracle Database
- Oracle Multimedia: Not available in Autonomous Database and deprecated in Oracle Database 18c.
- Oracle Sharding
- Custom locale objects, including: language, territory, character set, and collation (linguistic sort) are not supported. Custom locale definitions created using Oracle Locale Builder cannot be deployed on Autonomous Database. See Customizing Locale Data for more information.

Logical Partition Change Tracking and Materialized Views

Describes information about the Logical Partition Change Tracking (LPCT) metadata framework and Query Rewrite with Logical Partition Change Tracking in Autonomous Database.

- About Logical Partition Change Tracking
- Using Logical Partition Change Tracking
  Logical Partition Change Tracking (LPCT) logically partitions a table using a specified key column and method.
- Example: Logical Partition Change Tracking
  Shows the steps to use Logical Partition Change Tracking (LPCT) using a Materialized View that contains joins and aggregates.

About Logical Partition Change Tracking

Logical Partition Change Tracking (LPCT) tracks the staleness of materialized views.

Logical Partition Change Tracking enables you to create logical partitions on base tables. It evaluates the staleness of the base tables for individual logical partitions without using a materialized view log or requiring any of the tables used in the materialized view to be partitioned.

When one or more dependent base tables of a materialized view are updated, a materialized view becomes `STALE` and cannot be used for query rewrite under the default enforced integrity mode.

Logical Partition Change Tracking (LPCT) provides the capability to leverage the user-supplied logical partitioning information of base tables of a materialized view for a more fine-grained, partition-level tracking of stale data for both refresh and rewrite purposes. While classical Partitioning Change Tracking (PCT) relies on the physical partitioning of tables, LPCT has no dependency on tables being physically partitioned; LPCT can be used with both partitioned and nonpartitioned tables.

Logical Partition Change Tracking mechanism makes use of the `FRESH` subsets (partitions) of materialized views despite other subsets being `STALE`. Faster response times can be achieved for user queries because pre-computed results in materialized views are used more often. Other than increasing the usability of materialized views, PCT and LPCT also allows incremental refreshing of the materialized views without the need of materialized view logs; refresh can be both `ON DEMAND` or `ON COMMIT`.

Similar to Partitioning Change Tracking (PCT), Logical Partition Change Tracking (LPCT) is associated with a base table and can accurately identify the rows in a Materialized View affected by data changes on the base table, according to the logical partition boundaries defined.

See Advanced Materialized Views Advanced Materialized Views for more information.

Using Logical Partition Change Tracking

Logical Partition Change Tracking (LPCT) logically partitions a table using a specified key column and method.

Logical Partition Change Tracking creation syntax is analogous to physical partitions. Unlike physical partitions, which must be created as part of table creation, LPCT can be freely

specified independent of the table creation and its shape, allowing more flexibility to address your requirements. LPCT creation is metadata only.

- Creating Logical Partitions - BNF
  Describes the syntax to create BNF logical partitions.
- Choosing the Logical Partition Key Column
  The logical partitioning key is specified to define the boundaries of each logical partition.
- Freshness of Materialized Views Using Logical Partition Change Tracking
  The Logical Partition Change Tracking (LPCT) staleness tracking mechanism automatically records and consolidates the change statistics internally based on the specified logical partition key and partitioning method during each data change.
- Rewrite with Materialized Views Using Logical Partition Change Tracking
  Using Logical Partition Change Tracking (LPCT), Oracle knows that a materialized view is `STALE` with respect to some logical partitions of the base table, but `FRESH` with respect to other portions.
- Refresh of Materialized Views Using Logical Partition Change Tracking
  Logical Partition Change Tracking (LPCT) refresh can be implemented using the finer-grained data staleness to incrementally refresh `STALE` subsets of a materialized view, eliminating costly complete refresh or log-based fast refresh.
- Logical Partition Change Tracking – Data Dictionary Views
  Describes the data dictionary views to find information about logical partitions.

Creating Logical Partitions - BNF

Describes the syntax to create BNF logical partitions.

Following is the syntax to create BNF logical partitions:

```
CREATE LOGICAL PARTITION TRACKING ON table_name
    PARTITION BY RANGE (partition_key)
    INTERVAL (interval_clause)
    (partition_specification);
```

- Only `RANGE` and `INTERVAL` logical partitioning methods are supported.
- Only a single logical partition key column is supported.
- The partition key column can be of these data types:
  - `NUMBER`
  - `DATE`
  - `CHAR`
  - `VARCHAR`
  - `VARCHAR2`
  - `TIMESTAMP`
  - `TIMESTAMP WITH TIME ZONE`

Choosing the Logical Partition Key Column

The logical partitioning key is specified to define the boundaries of each logical partition.

The logical partition key is not physical, this means that table rows belonging to a key range are not segregated into a separate physical partition. The table can be non-partitioned or partitioned on a key that is different from the logical partition key. The logical partition key can be chosen freely, and partition boundaries can be made flexible.

To choose a Logical Partition Change Tracking (LPCT) key column, you can consider a clustered column, that is, a column where data is close to sorted by column value, that are frequently referenced in the query filter predicates. For a clustered column, less logical partitions are likely to be affected during data loads, this means that less `STALE` logical partitions need to be refreshed and more `FRESH` logical partitions are ready to be used for rewrites. If a table is already partitioned, it is recommended to create a LPCT using a different column other than the partition key column. LPCT offers similar benefits as Partitioning Change Tracking (PCT), and the combined benefits are not maximized if data tracking is done on the same column.

Freshness of Materialized Views Using Logical Partition Change Tracking

The Logical Partition Change Tracking (LPCT) staleness tracking mechanism automatically records and consolidates the change statistics internally based on the specified logical partition key and partitioning method during each data change.

Adjacent change data is grouped into a “logical” partition. Unlike with Partitioning Change Tracking (PCT), which is tied to physical partition bounds the LPCT scheme offers flexibility in managing and grouping the data changes resulting from DMLs applied to the base table.

During conventional DMLs and Direct-loads, LPCT adopts the same algorithm that PCT uses to track staleness. During Query rewrites, LPCT adopts the same algorithm that PCT uses to calculate rewrite containment.

When a table is logically partitioned using key ranges, a materialized view defined on the table is eligible to use LPCT for staleness tracking, refresh and query rewrite, provided that the materialized view contains the logical partition key.

> **Note:**
>
> All types of Materialized Views are supported for LPCT.

Rewrite with Materialized Views Using Logical Partition Change Tracking

Using Logical Partition Change Tracking (LPCT), Oracle knows that a materialized view is `STALE` with respect to some logical partitions of the base table, but `FRESH` with respect to other portions.

Having the finer-grained data staleness information of the base tables, the associated materialized view would be used more frequently due to LPCT rewrite.

Oracle transparently identifies and makes use of the `FRESH` subset of materialized views for query rewrite to answer complicated queries of base tables when `QUERY_REWRITE_INTEGRITY = ENFORCED |TRUSTED`.

If the materialized view rows are partially `FRESH` with respect to those logical partitions, a partial rewrite might take place to answer the query partially using materialized view, that is, `FRESH` logical partitions, and partially using the base table, that is, the `STALE` logical partitions.

Refresh of Materialized Views Using Logical Partition Change Tracking

Logical Partition Change Tracking (LPCT) refresh can be implemented using the finer-grained data staleness to incrementally refresh `STALE` subsets of a materialized view, eliminating costly complete refresh or log-based fast refresh.

If LPCT refresh is specified, the `STALE` logical partitions are identified and targeted refresh operations will be performed to those logical partitions only.

To invoke refresh using logical partition change tracking you specify `'L'` or `'l'` ("logical") as refresh method.

For example: `execute DBMS_MVIEW.REFRESH(<materialized_view_name>,'L');`

If `REFRESH FORCE` is specified, a `FAST` refresh is chosen and performed if possible, or else it performs a `COMPLETE` refresh. During materialized view `FORCE` refresh, LPCT refresh has the same priority as Partitioning Change Tracking (PCT) refresh.

Logical Partition Change Tracking – Data Dictionary Views

Describes the data dictionary views to find information about logical partitions.

Query the following data dictionary views to retrieve information about logical partitions.

- `ALL_MVIEW_DETAIL_LOGICAL_PARTITION`: This view displays the freshness information of the materialized views, with respect to an LPCT detail logical partition, accessible to the current user. See ALL_MVIEW_DETAIL_PARTITION for more information.
- `DBA_MVIEW_DETAIL_ LOGICAL_PARTITION`: displays freshness information for all materialized views in the database, with respect to a LPCT detail logical partition. See DBA_MVIEW_DETAIL_PARTITION for more information.
- `USER_MVIEW_DETAIL_ LOGICAL_PARTITION`: displays freshness information for all materialized views, with respect to a LPCT detail logical partition, owned by the current user. See USER_MVIEW_DETAIL_PARTITION for more information.

Example: Logical Partition Change Tracking

Shows the steps to use Logical Partition Change Tracking (LPCT) using a Materialized View that contains joins and aggregates.

1. Create base tables with logical change partitions.
   a. Create `MYSALES` table.

```
CREATE TABLE mysales ( time_id DATE, prod_id NUMBER, cust_id NUMBER,
channel_id CHAR(1), promo_id NUMBER(6), quantity_sold
      NUMBER(3), amount_sold NUMBER(10,2))
      PARTITION BY LIST (prod_id)
      (PARTITION p1 VALUES (1,2,3),
      PARTITION p2 VALUES (4,5,6),
      PARTITION p3 VALUES (7,8,9),
      PARTITION p4 VALUES(DEFAULT));
```

   This creates the `MYSALES` table.

b. Insert records into the `MYSALES` table.

```
INSERT INTO mysales (time_id, prod_id, cust_id, amount_sold) VALUES
(TO_DATE('2007-06-05','yyyy-mm-dd'), 1, 2088, 189.98);
INSERT INTO mysales (time_id, prod_id, cust_id, amount_sold) VALUES
(TO_DATE('2007-07-05','yyyy-mm-dd'), 2, 1354, 12.99);
INSERT INTO mysales (time_id, prod_id, cust_id, amount_sold) VALUES
(TO_DATE('2007-09-05','yyyy-mm-dd'), 5, 2088, 189.98);
INSERT INTO mysales (time_id, prod_id, cust_id, amount_sold) VALUES
(TO_DATE('2007-10-05','yyyy-mm-dd'), 18, 2088, 42);
COMMIT;
```

This populates the `MYSALES` table.

c. Create logical partition tracking for the `MYSALES` table.

```
CREATE LOGICAL PARTITION TRACKING ON mysales
  PARTITION BY RANGE (time_id)
  INTERVAL(NUMTOYMINTERVAL(2, 'YEAR'))
  (
   PARTITION p0 VALUES LESS THAN (TO_DATE('7-15-2005', 'MM-DD-YYYY')),
   PARTITION p1 VALUES LESS THAN (TO_DATE('7-15-2007', 'MM-DD-YYYY'))
  );
```

This creates a logical partition tracking for the `MYSALES` table using the `TIME_ID` as key.

d. Create `MYCUSTOMERS` table.

```
CREATE TABLE mycustomers (cust_id NUMBER, age NUMBER, gender CHAR(1),
address VARCHAR(100));
```

This creates the `MYCUSTOMERS` table.

e. Insert records into the `MYCUSTOMERS` table.

```
INSERT INTO mycustomers(cust_id, age, gender) VALUES (2088, 35, 'F');
INSERT INTO mycustomers(cust_id, age, gender) VALUES (1234, 54, 'M');
INSERT INTO mycustomers(cust_id, age, gender) VALUES (1354, 17, 'F');
INSERT INTO mycustomers(cust_id, age, gender) VALUES (6666, 15, 'F');
COMMIT;
```

This populates the `MYCUSTOMERS` table.

f. Create logical partition tracking for the `MYCUSTOMERS` table.

```
CREATE LOGICAL PARTITION TRACKING ON mycustomers
 PARTITION BY RANGE (age) INTERVAL (20.5)
 (PARTITION m0 values less than (20));
```

This creates a logical partition tracking for the `MYSALES` table using the `AGE` as key.

2. Create a materialized view on top of tables with logical partition change tracking.

a. Create materialized view on `MYSALES` and `MYCUSTOMERS` tables.

```
CREATE MATERIALIZED VIEW sales_age_time
 REFRESH FAST
 ENABLE QUERY REWRITE
 AS SELECT SUM(s.amount_sold) amount_total, c.age, s.time_id
 FROM mysales s, mycustomers c
 WHERE s.cust_id = c.cust_id
 GROUP BY c.age, s.time_id;
```

This creates the `SALES_AGE_TIME` materialized view.

b. Query the `DBA_MVIEW_DETAIL_LOGICAL_PARTITION` data dictionary view.

```
SELECT mview_name, DETAILOBJ_NAME, DETAIL_LOGICAL_PARTITION_NAME
LPARTNAME,
 DETAIL_LOGICAL_PARTITION_NUMBER LPART#, FRESHNESS
 FROM DBA_MVIEW_DETAIL_LOGICAL_PARTITION
 WHERE mview_name = 'SALES_AGE_TIME'
 ORDER BY 1,2,3;
```

It shows the following output.

```
MVIEW_NAME        DETAILOBJ_NAME     LPARTNAME      LPART#
FRESHNESS
---------------   ---------------    ----------     ----------
---------
SALES_AGE_TIME    MYCUSTOMERS        M0             0           FRESH
SALES_AGE_TIME    MYSALES            P0             0           FRESH
SALES_AGE_TIME    MYSALES            P1             1           FRESH
```

c. Use `EXPLAIN_MVIEW` to assess the logical partition related refresh and rewrite capabilities.

```
EXECUTE DBMS_MVIEW.EXPLAIN_MVIEW ('sales_age_time');

SELECT CAPABILITY_NAME, RELATED_TEXT, POSSIBLE
FROM MV_CAPABILITIES_TABLE
WHERE MVNAME = 'SALES_AGE_TIME' AND CAPABILITY_NAME LIKE '%LPT%'
ORDER BY 1, 2;
```

It shows the following output.

```
CAPABILITY_NAME              RELATED_TEXT                POSSIBLE
--------------------------   --------------------------
---------------
LPT                                                       Y
LPT_TABLE                    MYCUSTOMERS                  Y
LPT_TABLE                    MYSALES                      Y
LPT_TABLE_REWRITE            MYCUSTOMERS                  Y
LPT_TABLE_REWRITE            MYSALES                      Y
```

```
REWRITE_LPT                                                    Y
REFRESH_FAST_LPT                                               Y
```

3. Observe the impact of DMLs on your materialized view.

   a. Introduce a new logical partition on the `MYSALES` table.

   ```
   INSERT INTO mysales (time_id, prod_id, cust_id, amount_sold) VALUES
         (TO_DATE('2019-02-05','yyyy-mm-dd'), 99, 2108, 33);
   ```

   This introduces a new partition (partition #6) on the `MYSALES` table.

   b. Introduce a new logical partition on the `MYSALES` table.

   ```
   INSERT INTO mysales (time_id, prod_id, cust_id, amount_sold) VALUES
   (TO_DATE('2019-02-05','yyyy-mm-dd'), 99, 2108, 33);
   ```

   This introduces a new partition (partition #6) on the `MYSALES` table.

   c. Introduce a new logical partition on the `MYCUSTOMERS` table.

   ```
   INSERT INTO mycustomers(cust_id, age, gender) VALUES (1399, 80, 'F');
   ```

   This introduces a new partition (partition #3) on the `MYCUSTOMERS` table.

   d. Introduce a new logical partition on the `MYSALES` table.

   ```
   INSERT INTO mysales (time_id, prod_id, cust_id, amount_sold) VALUES
   (TO_DATE('2019-02-09','yyyy-mm-dd'), 99, 1997, 79.9);
   ```

   This introduces a new partition (partition #7) on the `MYSALES` table.

   e. Introduce a new logical partition on the `MYSALES` table.

   ```
   INSERT INTO mysales (time_id, prod_id, cust_id, amount_sold) VALUES
   (TO_DATE('2010-02-09','yyyy-mm-dd'), 110, 1997, 108.98);
   COMMIT;
   ```

   This introduces a new partition (partition #2) on the `MYSALES` table.

   f. Query the `DBA_MVIEW_DETAIL_LOGICAL_PARTITION` data dictionary view.

   ```
   SELECT mview_name, DETAILOBJ_NAME, DETAIL_LOGICAL_PARTITION_NAME
   LPARTNAME,
    DETAIL_LOGICAL_PARTITION_NUMBER LPART#, FRESHNESS
    FROM DBA_MVIEW_DETAIL_LOGICAL_PARTITION
    WHERE mview_name = 'SALES_AGE_TIME'
    ORDER BY 1,2,3;
   ```

   Now, it shows the following output.

   ```
   MVIEW_NAME        DETAILOBJ_NAME     LPARTNAME      LPART#
   FRESHNESS
   ```

```
---------------   ---------------   ----------   ----------
---------
SALES_AGE_TIME    MYCUSTOMERS       M0           0           FRESH
SALES_AGE_TIME    MYCUSTOMERS       SYS_88904P3  3           STALE
SALES_AGE_TIME    MYSALES           P1           0           FRESH
SALES_AGE_TIME    MYSALES           P1           1           FRESH
SALES_AGE_TIME    MYSALES           SYS_88899P3  2           STALE
SALES_AGE_TIME    MYSALES           SYS_88899P7  6           STALE
```

g. Perform LPCT rewrite on subset of lpart #1 on the `MYSALES` and lpart #0 on the `MYCUSTOMERS` tables.

```
DELETE FROM rewrite_table;
DECLARE
 stmt varchar2(2000) := q'#select sum(s.amount_sold) amount_total,
 c.age, s.time_id
 FROM mysales s, mycustomers c
 WHERE s.cust_id = c.cust_id
 AND s.time_id < TO_DATE ('07-07-2007', 'MM-DD-YYYY')
 AND c.age < 18
 GROUP BY c.age, s.time_id#';
BEGIN
 dbms_mview.explain_rewrite (stmt,'sales_age_time');
END;
/
SELECT mv_name, sequence, pass, message FROM rewrite_table;
```

h. Query `REWRITE_TABLE` to verify the rewrites.

```
SELECT mv_name, sequence, pass, message FROM rewrite_table;
```

i. Execute the following query.

```
SELECT SUM(s.amount_sold) amount_total,
 c.age, s.time_id
 FROM mysales s, mycustomers c
 WHERE s.cust_id = c.cust_id
 AND s.time_id < TO_DATE ('07-07-2007', 'MM-DD-YYYY')
 AND c.age < 18
 GROUP BY c.age, s.time_id;
```

j. View the explain plan for the above query to verify the rewrites.

```
SELECT * FROM TABLE(dbms_xplan.display_cursor);


PLAN_TABLE_OUTPUT
_____________________________________________________
SQL_ID  ampuzk8tbp6df, child number 0
-------------------------------------
SELECT SUM(s.amount_sold) amount_total,
c.age, s.time_id
FROM mysales s, mycustomers c
WHERE s.cust_id = c.cust_id
AND s.time_id < TO_DATE ('07-07-2007', 'MM-DD-YYYY')
```

```
AND c.age < 18
GROUP BY c.age, s.time_id;

Plan hash
        value: 3902795718

-------------------------------------------------------------------------
-----------------------
        | Id  | Operation                    | Name           | Rows  |
Bytes | Cost
        (%CPU)| Time

|-------------------------------------------------------------------------
-----------------------
        |   0 | SELECT STATEMENT             |                |
          |       |     2 (100)|
        ||*  1 |  MAT_VIEW
        REWRITE ACCESS FULL| SALES_AGE_TIME |     1 |    35 |
          2   (0)| 00:00:01

|-------------------------------------------------------------------------
-----------------------
        Predicate Information (identified by operation id):
        ---------------------------------------------------
        1 - filter(("SALES_AGE_TIME"."TIME_ID"<TO_DATE('2007-07-07
00:00:00', 'syyyy-mm-dd hh24:mi:ss') AND "SALES_AGE_TIME"."AGE"<18))
Note
-----
   - dynamic statistics used: dynamic sampling (level=2)
26 rows selected.
```

4. Leverage LPCT for incremental refresh.
   a. Execute the following code to perform LPCT refresh.

```
EXECUTE DBMS_MVIEW.REFRESH('SALES_AGE_TIME', 'L');
```

   b. Verify the refresh using the following query.

```
SELECT mview_name, DETAILOBJ_NAME, DETAIL_LOGICAL_PARTITION_NAME
LPARTNAME, DETAIL_LOGICAL_PARTITION_NUMBER LPART#, FRESHNESS
 FROM DBA_MVIEW_DETAIL_LOGICAL_PARTITION
 WHERE mview_name = 'SALES_AGE_TIME'
 ORDER BY 1,2,3;
```

Migrating MySQL and Third-Party Databases to Autonomous Database

Migration is the process of copying the schema objects and data from a source MySQL or a third-party (non-Oracle) database, such as Amazon Redshift, Microsoft SQL Server, Sybase Adaptive Server, IBM DB2 (UDB), or Teradata, to Autonomous Database.

Using Oracle SQL Developer Migration Wizard you can perform migration in an efficient and largely automated way.

- Migrating Amazon Redshift to Autonomous Database
  The SQL Developer Amazon Redshift Migration Assistant, available with SQL Developer 18.3 and later versions provides a framework for easy migration of Amazon Redshift environments on a per-schema basis.
- Migrating MySQL to Autonomous Database
  Migration is the process of copying the schema objects and data from a source MySQL to Autonomous Database. Using Oracle SQL Developer Migration Wizard you can perform migration in an efficient and largely automated way.
- Migrating Microsoft SQL Server or Sybase Adaptive Server to Autonomous Database
  Migration is the process of copying the schema objects and data from a source Microsoft SQL Server or Sybase Adaptive Server to Autonomous Database. Using Oracle SQL Developer Migration Wizard you can perform migration in an efficient and largely automated way.
- Migrating IBM DB2 (UDB) Database to Autonomous Database
  Migration is the process of copying the schema objects and data from a source IBM DB2 (UDB) database to Autonomous Database. Using Oracle SQL Developer Migration Wizard you can perform migration in an efficient and largely automated way.
- Migrating Teradata Database to Autonomous Database
  Migration is the process of copying the schema objects and data from a source Teradata database to Autonomous Database. Using Oracle SQL Developer Migration Wizard you can perform migration in an efficient and largely automated way.

Migrating Amazon Redshift to Autonomous Database

The SQL Developer Amazon Redshift Migration Assistant, available with SQL Developer 18.3 and later versions provides a framework for easy migration of Amazon Redshift environments on a per-schema basis.

This section describes the steps and the workflow for both an online migration of Amazon Redshift and for the generation of scripts for a scheduled, manual migration that you run at a later time.

- Autonomous Database Redshift Migration Overview
  Using SQL Developer you can migrate database files from Amazon Redshift to Autonomous Database.
- Connect to Amazon Redshift
  Using SQL Developer you can migrate database Schemas and Tables from Amazon Redshift to Autonomous Database. To extract metadata and data from Amazon Redshift for a migration, you need to connect to Amazon Redshift with SQL Developer.
- Connect to Autonomous Database
  Using SQL Developer you create a connection to Autonomous Database
- Start the Cloud Migration Wizard
  Invoke the Cloud Migration Wizard from the Tools menu of SQL Developer to initiate your Amazon Redshift migration to Autonomous Database.
- Review and Finish the Amazon Redshift Migration
  The summary shows a summary of the information that you have specified.
- Use Generated Amazon Redshift Migration Scripts
  When you choose to generate migration scripts a new subdirectory is created in the local directory specified in the migration Wizard. You can run these scripts in real time or use them for programmatic processing.

- Perform Post Migration Tasks
  After successful migration of your Redshift environment you should consider the following post-migration tasks:

Autonomous Database Redshift Migration Overview

Using SQL Developer you can migrate database files from Amazon Redshift to Autonomous Database.

- **Capture:** Captures Metadata schemas and tables from source database and stores in Migration Repository.
- **Convert:** Redshift Datatypes are mapped to Oracle Datatypes. Redshift Object names are converted to Oracle names based on Oracle Naming Convention. The Column Defaults that use Redshift functions are replaced by their Oracle equivalents.
- **Generate:** Generate schemas and DDLs based on the converted metadata.
- **Deploy:** Deploy the generated schemas and DDLs.
- **Copy Data:** Unload data from Redshift tables to Amazon Storage S3 then copy data from Amazon Storage to Autonomous Database tables(in schemas) that were Deployed earlier.

Connect to Amazon Redshift

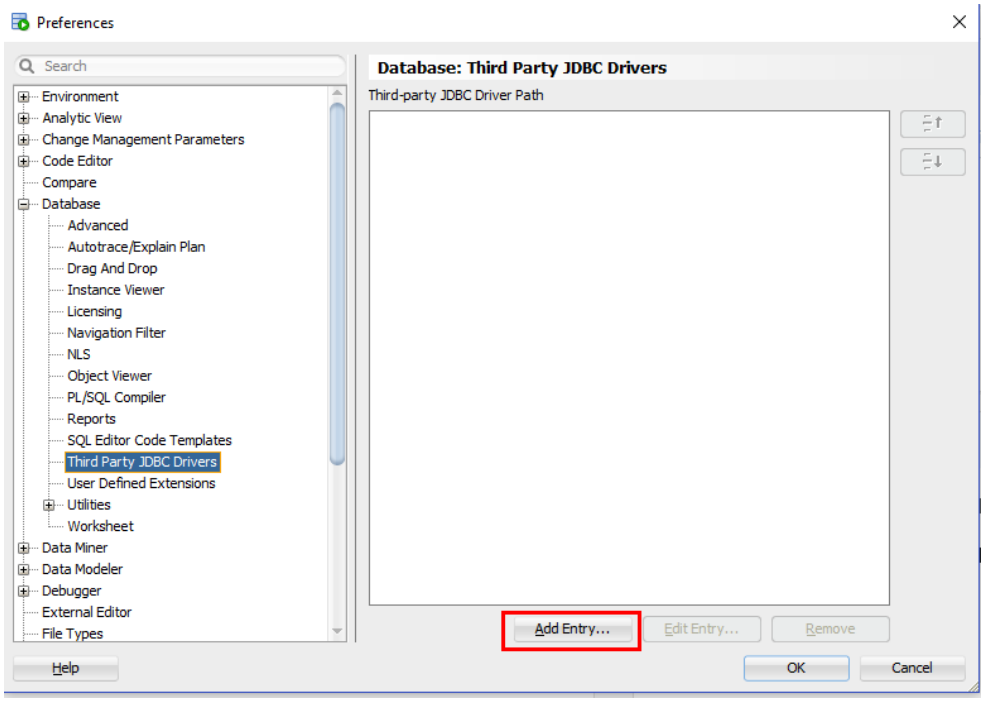
Using SQL Developer you can migrate database Schemas and Tables from Amazon Redshift to Autonomous Database. To extract metadata and data from Amazon Redshift for a migration, you need to connect to Amazon Redshift with SQL Developer.

Download Amazon Redshift JDBC Driver and Add the Third Party Driver

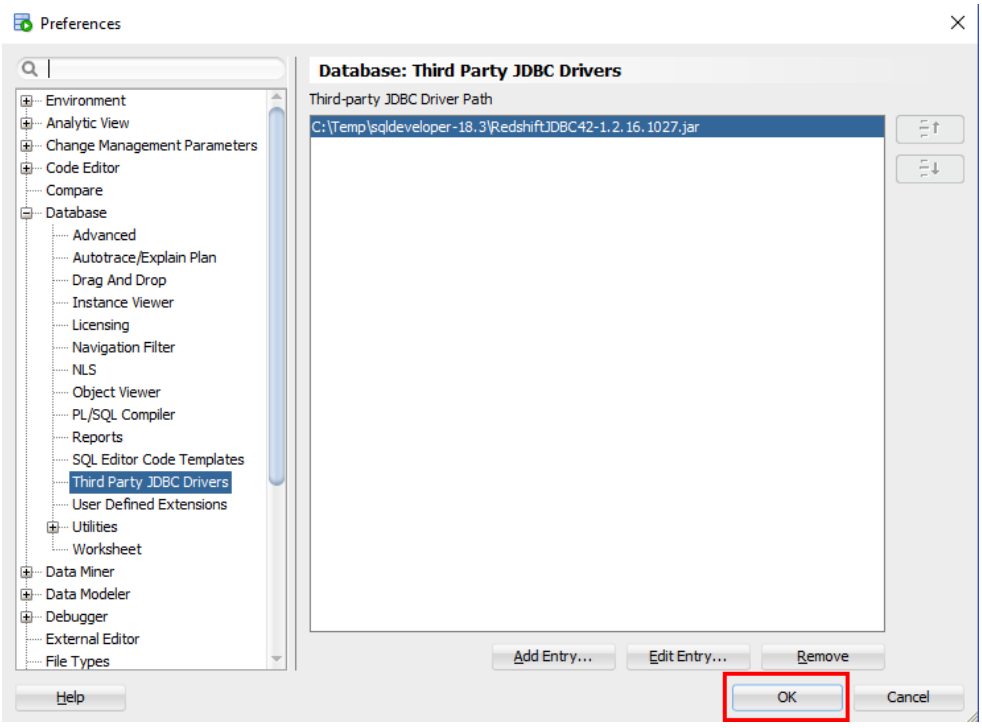
1. Download an Amazon Redshift JDBC driver to access Amazon Redshift. Consult the Amazon Redshift documentation for the location of the most recent JDBC driver. For more information, see Configure a JDBC Connection.

   > **Note:**
   >
   > Use the Amazon Redshift JDBC Driver JDBC 4.2–compatible driver.

2. Store the Amazon Redshift JDBC driver in a local directory where SQL Developer can access the Amazon Redshift JDBC driver.
3. Add the Amazon Redshift JDBC driver as third party to SQL Developer before making a connection. Within SQL Developer, go to **Tools > Preferences > Database > Third Party JDBC Drivers** (for Mac, this is **Oracle SQL Developer > Preferences Database > Third Party JDBC Drivers**).
4. Click **Add Entry** and select the path to the Amazon Redshift JDBC Driver that you download.



5. Click **OK** to add the Amazon Redshift JDBC driver that you download.



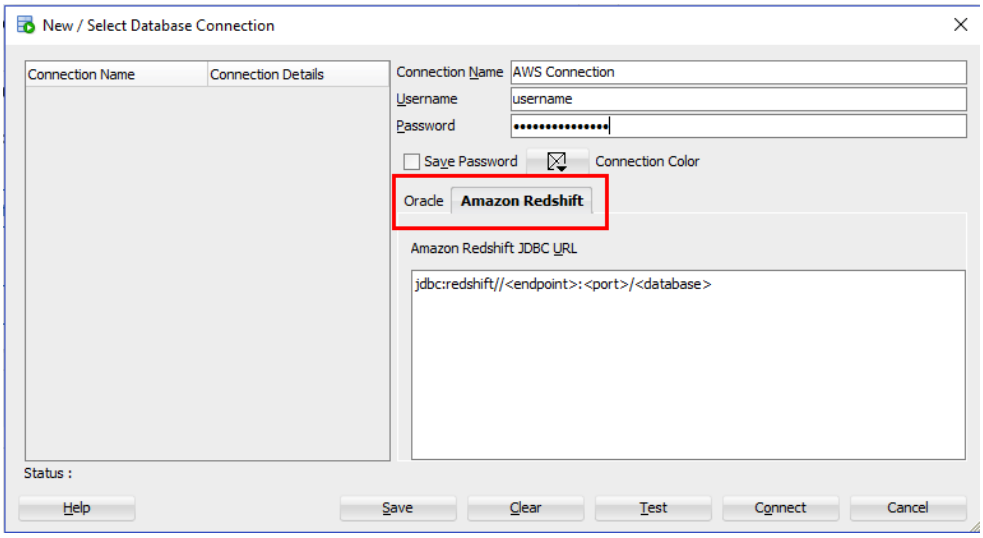
Add Connection to Amazon Redshift Database

Connect to the Amazon Redshift database.

1. In the Connections panel, right-click **Connections** and select **New Connection...**.

2. Select the **Amazon Redshift** tab and enter the connection information for Amazon Redshift.

   If you are planning to migrate multiple schemas it is recommended to connect with the Master username to your Amazon Redshift system.



- For more details for configuring a JDBC Connection and obtaining the Amazon Redshift JDBC URL, see AWS: Configure a JDBC Connection.
- For more details for configuring security options for the connection (in case of "Amazon [500150] connection error"), see AWS: Configure Security options for Connection (in case of "Amazon [500150] connection error").
- If you deployed your Amazon Redshift environment within a Virtual Private Cloud (VPC) you have to ensure that your cluster is accessible from the Internet. See http://docs.aws.amazon.com/redshift/latest/gsg/rs-gsg-authorize-cluster-access.html for details of how to enable public Internet access.
- If your Amazon Redshift client connection to the database appears to hang or times out when running long queries, see http://docs.aws.amazon.com/redshift/latest/mgmt/connecting-firewall-guidance.html for possible solutions to address this issue.

Test the connection before you save it.

Additional Information for Amazon Authentication and Access Control

- AWS: Security
- AWS: Managing Cluster Security Groups Using the Console

Connect to Autonomous Database

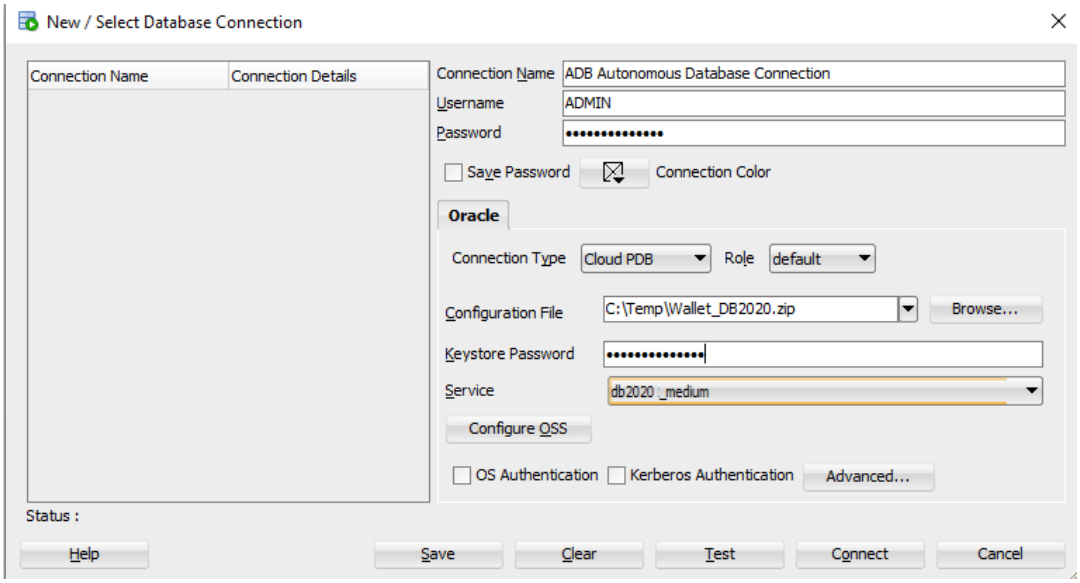
Using SQL Developer you create a connection to Autonomous Database

Obtain the client credentials wallet zip file. For more information, see Download Client Credentials (Wallets).

1. In the Connections panel, right-click **Connections** and select **New Connection...**.
2. Select the **Oracle** tab and enter the connection information for Autonomous Database.
3. For the AWS Redshift Migration connection, select the **_low** connection to your database.

For more information, see Predefined Database Service Names for Autonomous Database.

4. Add a connection to Autonomous Database.



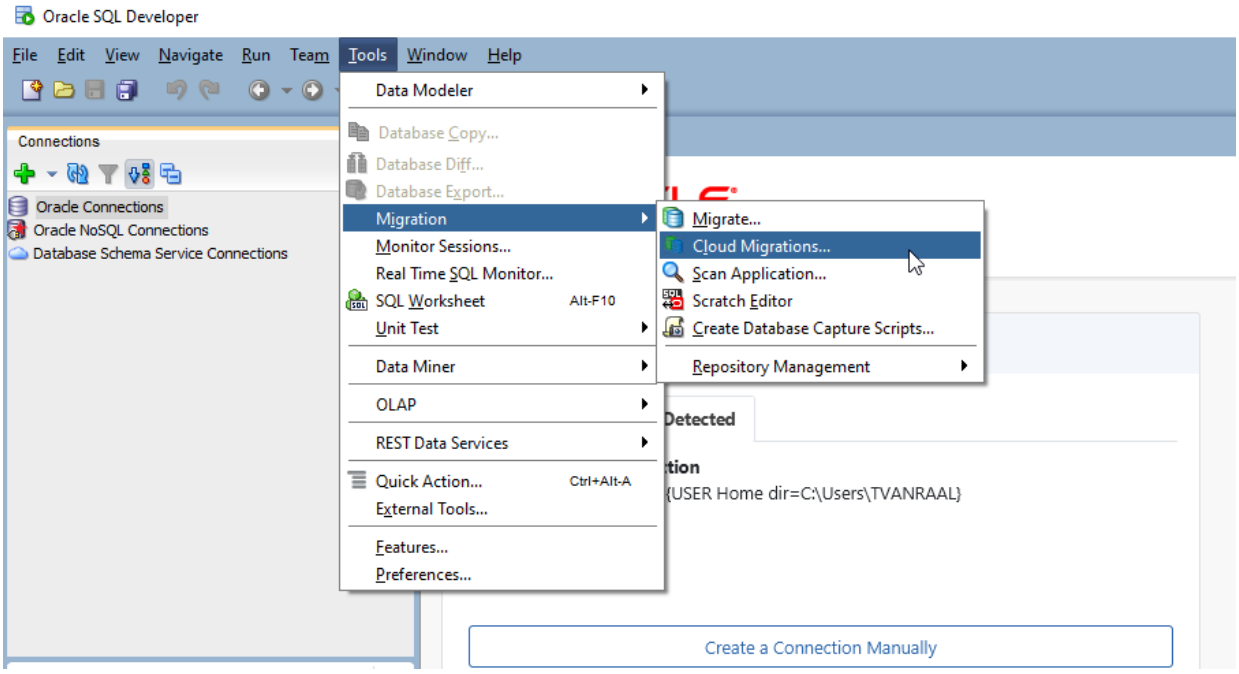
See Connect Oracle SQL Developer with a Wallet (mTLS) for more information.

Test the connection before you save it.

Start the Cloud Migration Wizard

Invoke the Cloud Migration Wizard from the Tools menu of SQL Developer to initiate your Amazon Redshift migration to Autonomous Database.

The Cloud Migration wizard starts when you click Cloud Migrations from **Migration** in the **Tools** menu. The Cloud Migrations wizard enables you to migrate schemas, objects (tables), and data from an Amazon Redshift database to Autonomous Database.



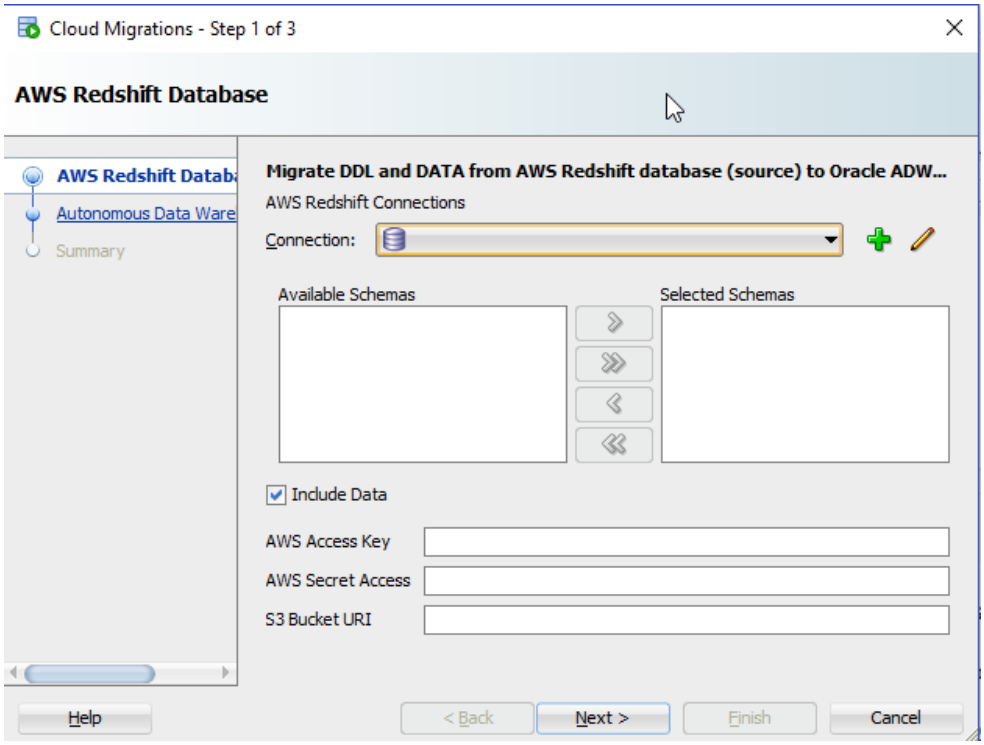
The Cloud Migration Wizard is an easy set of steps. The Cloud Migration Wizard guides you to:

- Identify the schemas in your Amazon Redshift database that you want to migrate.
- Identify the target Autonomous Database.
- Define whether you want to migrate the metadata (DDL), the data, or both.
- Choose to migrate your system online, to generate scripts, or both.

Identify the Amazon Redshift Database

Identify the schemas in the Amazon Redshift database to migrate. All objects, mainly tables, in the schema will be migrated. Migration to Autonomous Database is on a per-schema basis. Schemas cannot be renamed as part of the migration.

1. In the AWS Redshift Migration, specify the Connection.



- **Connection:** Name of the Redshift database connection.
- **Available Schemas:** Schemas available for the specific connection.
- **Selected Schemas:** Click the Add icon to select the schemas you want to migrate from **Available Schemas**.
- **Include Data:** DDL and DATA migrates the selected schemas and data.

When you migrate data, you have to provide the AWS access key, AWS Secret Access Key, and an existing S3 bucket URI where the Redshift data will be unloaded and staged. The security credentials require certain privileges to store data in S3. It is recommended to create new, separate access keys for the migration. The same access key is used later on to load the data into the Autonomous Database using secure REST requests.

- **AWS Access Key:** For more information on access keys, see AWS Identity and Access Management.
- **AWS Secret Access:** For more information on access keys, see AWS Identity and Access Management.
- **S3 Bucket URI:** For information on common S3 ServiceException errors, see S3ServiceException Errors.

  For more information an S3 buckets, see Creating and Configuring an S3 Bucket.

Amazon S3 Bucket URI Format

For the source files that reside in Amazon S3, see the following for a description of the URI format for accessing your files: Accessing a Bucket For example the following refers to the file folder 'folder_name' in the adwc bucket in the us-west-2 region.

```
https://s3-us-west-2.amazonaws.com/adwc/folder_name
```

S3 Bucket Configuration Example 1

If you provide the following S3 Bucket URI :

```
https://s3-us-west-2.amazonaws.com/my_bucket
```

The wizard verifies the entire path including *my_bucket*. An attempt is made to write a test file, if it is not accessible there is a prompt. In case, `my_bucket` does not exist, there is an error reported:

```
Validation Failed
```

Then the code generation creates the following path for the `DBMS_CLOUD.COPY_DATA` function:

```
file_uri_list => "https://s3-us-west-2.amazonaws.com/my_bucket/
oracle_schema_name/oracle_table_name/*.gz"
```

The migration assistant creates these folders: *oracle_schema_name*/*oracle_table_name* inside the bucket: *my_bucket*.

S3 Bucket Example 2

If you provide the following S3 Bucket URI :

```
https://s3-us-west-2.amazonaws.com/my_bucket/another_folder
```

The wizard verifies the entire path including *my_bucket*. An attempt is made to write a test file, if it is not accessible there is a prompt. In case, `my_bucket` does not exist, there is an error reported:

```
Validation Failed
```

In this case the *another_folder* does not have to exist. The migration creates the *another_folder* bucket inside *my_bucket*.

Then the code generation creates the following path for the `DBMS_CLOUD.COPY_DATA` function:

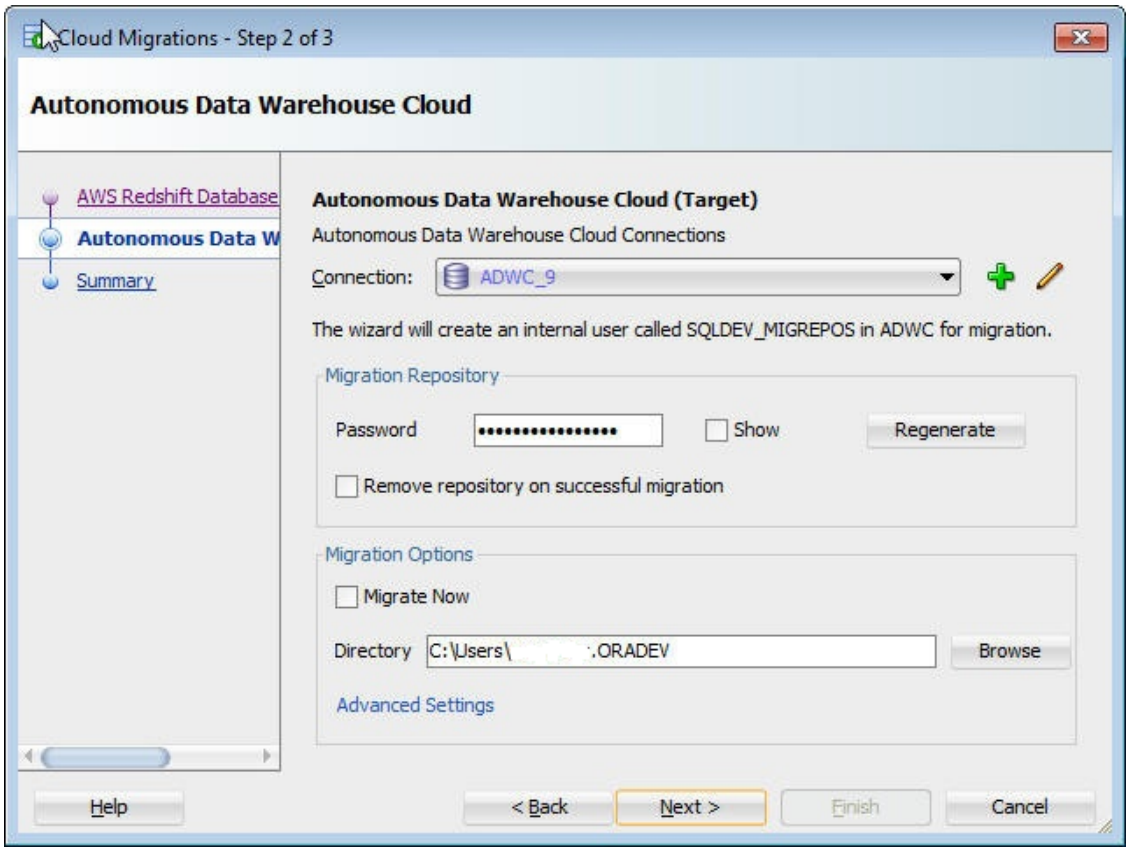
```
file_uri_list => 'https://s3-us-west-2.amazonaws.com/my_bucket/another_folder/
oracle_schema_name/oracle_table_name/*.gz
```

Step 2 of 3: Autonomous Data Warehouse Cloud

First create a connection for your target Autonomous Database See Connect to Autonomous Database. The user for this connection must have the administrative privileges; the connection is used throughout the migration to create schemas and objects.

> **Note:**
>
> Use the ADMIN user or a user with admin role.

The Amazon Redshift Migration Assistant allows you to do an online migration right away, to generate all scripts necessary for a migration, or both. If you chose to store the scripts in a local directory you have to specify the local directory (the directory must be writable by the user).



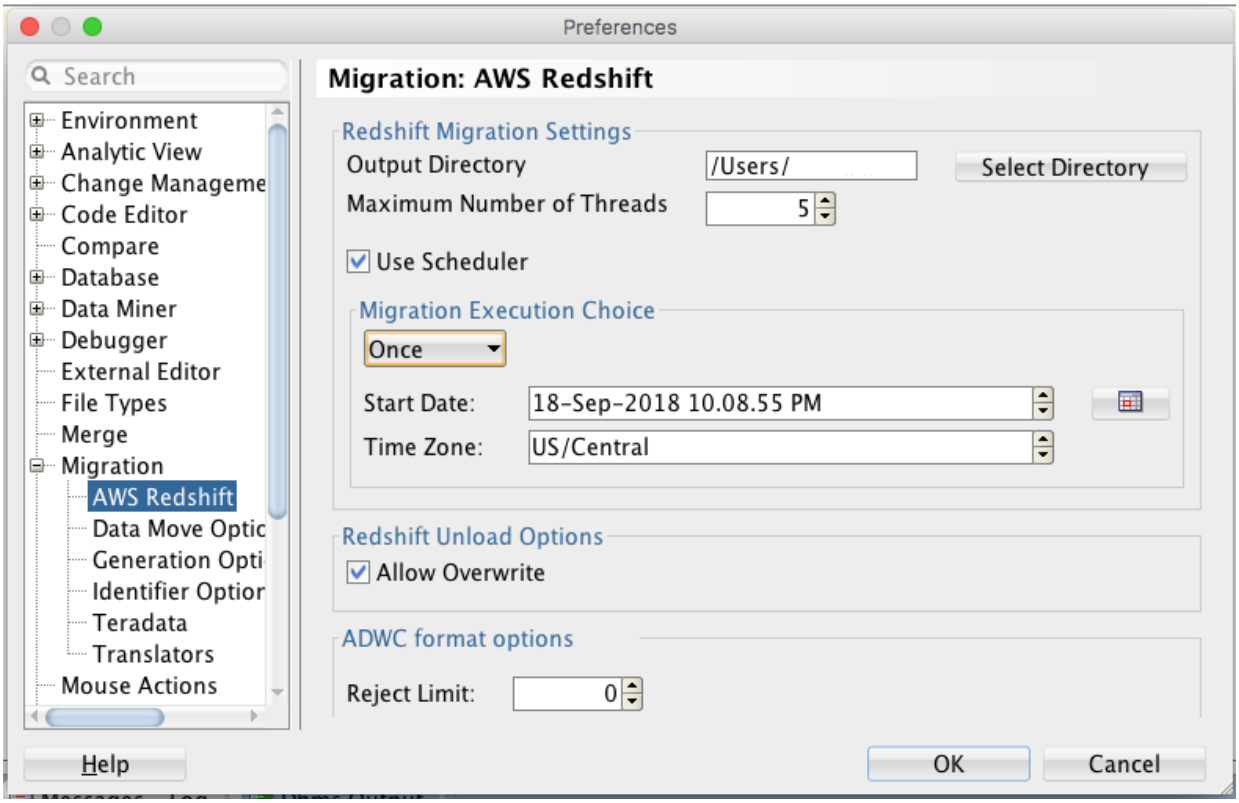
- **Connection:** Name of the Autonomous Data Warehouse Cloud connection. Create a connection for the Autonomous Database if required. The user must have administrative privileges since this connection is used throughout the migration to create schemas and objects.
- **Migration Repository Password:** Password for the migration repository that is installed in the Autonomous Database as part of the schema migration. Either use the pre-filled password or enter a new password.
- **Remove repository on successful migration:** Select this option to remove the repository after the migration is completed. The repository is not required after migration.
- **Migrate Now:** Select this option to perform an online migration immediately.

> **Note:**
>
> – If **Include Data** from Step 1 and **Migrate Now** are both unselected, you are opting for just generation of all required SQL Scripts for manual migration.
>
> – If **Include Data** from Step 1 is unchecked and **Migrate Now** is selected, then all selected schemas and their tables will be deployed in Autonomous Database but data will not be loaded into tables.
>
> – If **Include Data** from Step 1 and **Migrate Now** are both selected, then all selected schemas and their tables will be deployed in Autonomous Database and data will be loaded into tables.

- **Directory:** Specify the director to store the generated scripts necessary for the migration; this saves the scripts in a local directory.

Advanced Settings (Optional)

The default settings should work unless you want to control the format options when Unloading to S3 storage or Copying from S3 storage to Autonomous Database. For more information on Format Options, see DBMS_CLOUD Package Format Options. To use advanced options, click **Advanced Settings**.



Output Directory: Enter the path or click Select Directory to select the directory or folder for the migration.

Maximum Number of Threads: Enter the number of parallel threads to enable when loading data to tables in Autonomous Database.

Use Scheduler: Select this option to enable the scheduler for migration. You can schedule jobs for data load migration operations from the AWS S3 bucket to Autonomous Database. You have the option to run the scheduled jobs immediately or at a future date and time. To monitor the data load scheduled jobs, use the Scheduler node in the Connections navigator.

Migration Execution Choice:

- **Immediate** runs the scheduler as soon as the Redshift migration is triggered.
- **Once** runs the scheduler on a future date. You specify the **Start Date** and **Time Zone**. By default, the Start Date displays the current date and time of the local system. To change the start date, use the calendar icon to double-click and select the date or use the spinner to highlight the date and then click the field to set it.

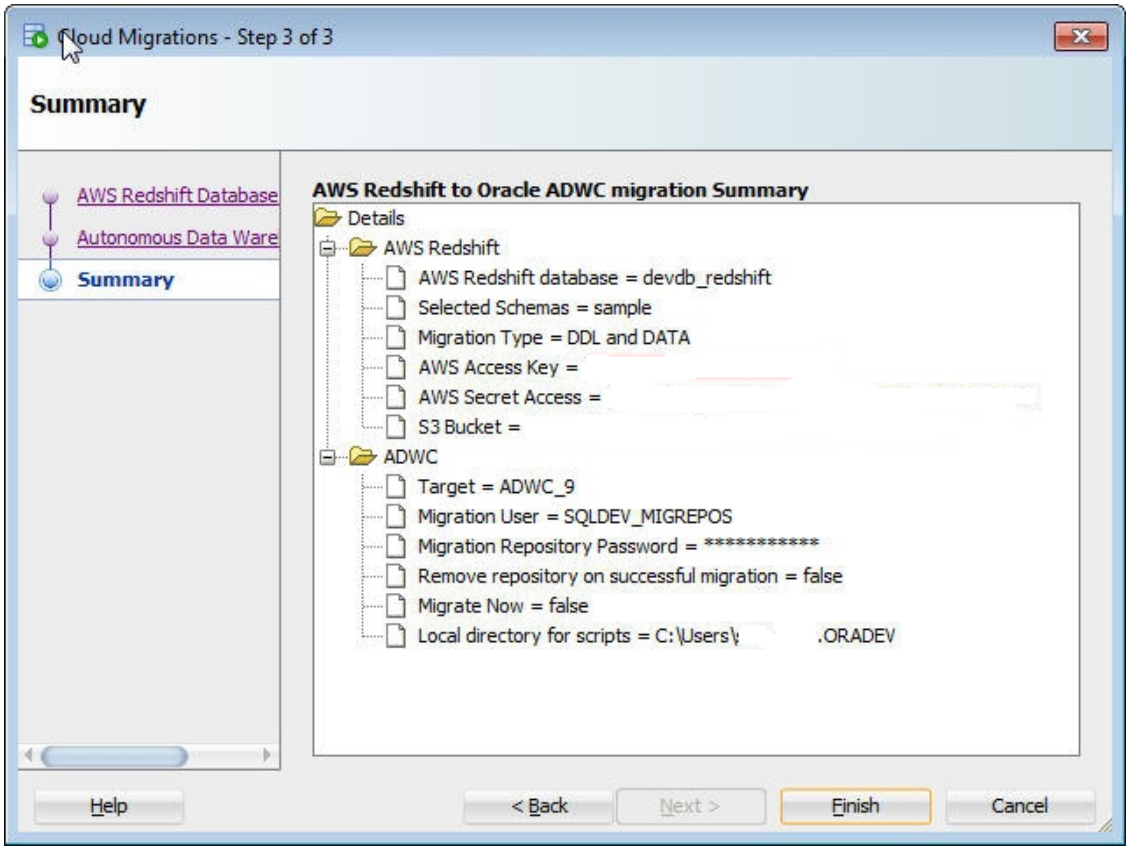
Redshift Unload Options: Allow Overwrite: If this option is enabled, the unload process will overwrite existing files, including the manifest file (lists the data files that are created by the unload process). By default, unload fails if there are files that can be overwritten.

ADWC format options: Reject Limit: Enter the number of rows to reject when loading data to tables in Autonomous Database. The migration operation will error out after the specified number of rows are rejected. The default is 0.

Review and Finish the Amazon Redshift Migration

The summary shows a summary of the information that you have specified.

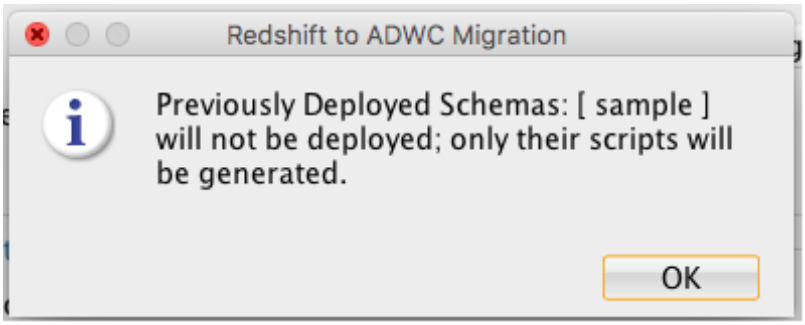
To change any information, press **Back** as needed.



If you have chosen an immediate migration, then the dialog of the migration wizard stays open until the migration is finished. If you select generate scripts, the migration process generates the necessary scripts in the specified local directory, and does not run the scripts.

To perform the migration, click **Finish**

If the selected schema name in AWS Redshift already exists in Autonomous Database, the migration process excludes deploying these selected schemas and displays a dialog:



Summary: What The Migration Assistant Creates

- Creates a new Autonomous Database user using the *schema_name* from Redshift.
- Creates a new bucket on S3 based on the *schema name*.
- Creates sub-folders on S3 for each table.

Use Generated Amazon Redshift Migration Scripts

When you choose to generate migration scripts a new subdirectory is created in the local directory specified in the migration Wizard. You can run these scripts in real time or use them for programmatic processing.

The directory contains the following scripts:

- `redshift_s3unload.sql`
- `adwc_ddl.sql`
- `adwc_dataload.sql`
- `adwc_dataload_scheduler.sql`

These scripts contain all necessary commands to migrate your Amazon Redshift system to Autonomous Database. You can run these scripts in real time or use them for programmatic processing.

Unload Your Amazon Redshift Data into S3

The first step of a successful migration is to unload your Amazon Redshift data into Amazon S3, which acts as a staging area. Script **redshift_s3unload.sql** has all the Amazon Redshift unload commands to unload the data using the access credentials and the S3 bucket that were specified in the Migration Wizard workflow.

Connect to your Amazon Redshift environment to run this script.

Create Your Data Warehouse Objects

To prepare your Autonomous Database create your empty data warehouse schema prior to loading data. The Amazon Redshift Migration Assistant converted all Amazon Redshift schema structures into Oracle structures in script **adwc_ddl.sql**.

The script must be executed while you are connected to your Autonomous Database as privileged user; for example, ADMIN.

By default, the schema created for the migration has the same name as the schema in Amazon Redshift. You must change the password to the valid password for the specified user either in the script or after the script runs. If you want to change the schema name then change the schema name and all references to the name.

Load Your Amazon Redshift Data into Your Oracle Autonomous Database

The script **adwc_dataload.sql** contains all the load commands necessary to load your unloaded Amazon Redshift data straight from S3 into your Autonomous Database.

Execute the script while connected to your Autonomous Database as a privileged user; for example ADMIN.

If you want to change the target schema name when you create your data warehouse objects then you must adjust the target schema names in this script accordingly.

Use of JOB SCHEDULER

SQL Developer provides a graphical interface for using the DBMS_SCHEDULER PL/SQL package to work with Oracle Scheduler objects. To use the SQL Developer scheduling features, please refer 'Scheduling Jobs Using SQL Developer' topic of SQL Developer User Guide and **Oracle Database Administrator's Guide** to understand the concepts and essential tasks for job scheduling.

The Scheduler node for a connection appears in the Connections navigator and in the DBA navigator. Use ADWC 'admin' user to navigate which displays Scheduler objects owned by the 'admin' monitoring status of data load jobs.

Under ADWC 'admin' Connection → Scheduler → Jobs, you will see AWS Redshift to ADWC data load jobs are created with name <schema_name>_<table_name>.

To see the status of completion of each data load, please expand each scheduled job and check the status.

Also for more detailed information about data load operation see table `MD_REPORT` in `SQLDEV_MIGREPOS` schema that stores information about table columns: and

```
 OPERATION_ID, LOGFILE_TABLE, BADFILE_TABLE, SOURCE_SCHEMA_NAME,
TARGET_SCHEMA_NAME, SOURCE_TABLE_NAME,
```

and

```
 TARGET_TABLE_NAME, SOURCE_TABLE_ROWS, TARGET_TABLE_ROWS_LOADED, ERROR
MESSAGE,
```

and

```
STATUS (COMPLETED or FAILED)
```

Redshift Migration Log and Report Files

After Redshift Migration, you will find three files:

- `MigrationResults.log` : Log file of Redshift migration
- `readme.txt` : file explains how to use the Generated Amazon Redshift Migration Scripts.
- `redshift_migration_reportxxx.txt` : Contains information about Migration, here is sample:

```
_____________________________________________________________
OPERATION ID : 8566
LOGFILE TABLE : COPY$8566_LOG
BADFILE TABLE : COPY$8566_BAD
SOURCE SCHEMA : sample
TARGET SCHEMA : SAMPLE
SOURCE TABLE : listing
TARGET TABLE : LISTING
SOURCE TABLE ROWS : 192497
TABLE ROWS LOADED : 192497
ERROR MESSAGE : null
STATUS : COMPLETED
START TIME : 2018-09-27 17:25:18.662075
END TIME : 2018-09-27 17:25:25.012695
_____________________________________________________________
```

Perform Post Migration Tasks

After successful migration of your Redshift environment you should consider the following post-migration tasks:

- Drop schema `SQLDEV_MIGREPOS`
- Drop the Amazon S3 bucket used for staging
- Harden the Amazon account used for accessing S3
- Drop the database credential used for data loading from S3
- Harden your accounts in your Autonomous Database

1. Drop schema `SQLDEV_MIGREPOS`

   As part of the schema migration the Migration Assistant installs a minimal migration repository in the target Autonomous Database. After the migration this account is no longer needed and can be dropped or alternatively locked.

2. Drop the Amazon S3 Bucket Used for Staging

   Unless you desire to use the unloaded Redshift data otherwise you can drop the bucket containing the unloaded data.

3. Harden the Amazon Account Used for Accessing S3

   You should inactivate the security access key used for S3 access unless needed for other purposes.

4. Drop the database credential used for data loading from S3

   The Amazon security credentials to access S3 are stored encrypted as database credential `REDSHIFT_DWCS_CREDS` in your Autonomous Database in the privileged user

schema that was used for the migration. Oracle recommends you drop this credential after successful migration unless needed for other purposes. For more information, see DROP_CREDENTIAL Procedure.

5. Harden your Accounts in Your Autonomous Database

   For the new schema created as part of the migration with the Migration Assistant, ensure to change the passwords of these accounts or lock and expire them if they're solely used for data storage.

Migrating MySQL to Autonomous Database

Migration is the process of copying the schema objects and data from a source MySQL to Autonomous Database. Using Oracle SQL Developer Migration Wizard you can perform migration in an efficient and largely automated way.

See SQL Developer: Migrating Third-Party Databases and Before Migrating From MySQL for more information.

Migrating Microsoft SQL Server or Sybase Adaptive Server to Autonomous Database

Migration is the process of copying the schema objects and data from a source Microsoft SQL Server or Sybase Adaptive Server to Autonomous Database. Using Oracle SQL Developer Migration Wizard you can perform migration in an efficient and largely automated way.

See SQL Developer: Migrating Third-Party Databases and Before Migrating From Microsoft SQL Server or Sybase Adaptive Server for more information.

Migrating IBM DB2 (UDB) Database to Autonomous Database

Migration is the process of copying the schema objects and data from a source IBM DB2 (UDB) database to Autonomous Database. Using Oracle SQL Developer Migration Wizard you can perform migration in an efficient and largely automated way.

See SQL Developer: Migrating Third-Party Databases and Before Migrating From IBM DB2 for more information.

Migrating Teradata Database to Autonomous Database

Migration is the process of copying the schema objects and data from a source Teradata database to Autonomous Database. Using Oracle SQL Developer Migration Wizard you can perform migration in an efficient and largely automated way.

See SQL Developer: Migrating Third-Party Databases and Before Migrating From Teradata for more information.

SODA Collection Metadata

It describes SODA Collection metadata on the database.

- SODA Collection Metadata on Autonomous Database
  Describes default and customized collection metadata on Autonomous Database.

SODA Collection Metadata on Autonomous Database

Describes default and customized collection metadata on Autonomous Database.

- SODA Default Collection Metadata on Autonomous Database
  Describes the default collection metadata on Autonomous Database, that is the metadata for a collection that is added when custom metadata is not supplied.
- SODA Customized Collection Metadata on Autonomous Database
  Describes SODA collection custom metadata on Autonomous Database.

SODA Default Collection Metadata on Autonomous Database

Describes the default collection metadata on Autonomous Database, that is the metadata for a collection that is added when custom metadata is not supplied.

Each SODA implementation provides a way to create a default collection when you supply a collection name. For example, in SODA for Java you use the `createCollection` method and supply just a collection name parameter:

```
db.admin().createCollection("myCol");
```

This creates a collection with default collection metadata. When you create a default collection on your database, the collection metadata includes the following information (regardless of which SODA implementation you use to create the default collection):

```
{
   "keyColumn" :
   {
      "name" : "ID",
      "sqlType" : "VARCHAR2",
      "maxLength" : 255,
      "assignmentMethod" : "UUID"
   },

   "contentColumn" :
   {
      "name" : "JSON_DOCUMENT",
      "sqlType" : "BLOB",
      "jsonFormat" : "OSON"
   },
   "versionColumn" :
   {
     "name" : "VERSION",
     "method" : "UUID"
   },

   "lastModifiedColumn" :
   {
     "name" : "LAST_MODIFIED"
   },

   "creationTimeColumn" :
   {
      "name" : "CREATED_ON"
```

```
   },

   "readOnly" : false
}
```

> **Note:**
>
> Using Always Free Autonomous Database with Oracle Database 21c, the default metadata changes as follows.

```
{
   "keyColumn" :
   {
      "name" : "ID",
      "sqlType" : "VARCHAR2",
      "maxLength" : 255,
      "assignmentMethod" : "UUID"
   },

   "contentColumn" :
   {
      "name" : "JSON_DOCUMENT",
      "sqlType" : "JSON",
   },
   "versionColumn" :
   {
     "name" : "VERSION",
     "method" : "UUID"
   },

   "lastModifiedColumn" :
   {
     "name" : "LAST_MODIFIED"
   },

   "creationTimeColumn" :
   {
      "name" : "CREATED_ON"
   },

   "readOnly" : false
}
```

SODA Customized Collection Metadata on Autonomous Database

Describes SODA collection custom metadata on Autonomous Database.

Each SODA implementation provides a way to customize the collection metadata during collection creation. For example, in SODA for Java, you can use the following command:

```
OracleDocument metadata = db.createDocumentFromString("metadata_string");
OracleCollection col = db.admin().createCollection("myCustomColl", metadata);
```

In this example, for *metadata_string* you can use the default metadata as the starting point, and customize the following:

- Change `keyColumn.assignmentMethod` to `CLIENT`: Change the value of the `assignmentMethod` under `keyColumn` in the metadata to `CLIENT` (instead of `UUID`).

  Valid values for `keyColumn.assignmentMethod` on Autonomous Database:

  - **UUID** (default): Keys are generated by SODA, based on the `UUID`.
  - **CLIENT**: Keys are assigned by the client application.

- Provide a `mediaTypeColumn name` value: A media type column is needed if the collection is to be heterogeneous, that is, it can store documents other than JavaScript Object Notation (JSON). See Media Type Column Name for details.

The following example specifies client-assigned keys and a custom media type column. The `mediaTypeColumn name` is specified with the value *YOUR_MEDIA_TYPE_COLUMN_NAME*. Otherwise, the default settings are used.

```
{
   "keyColumn" :
   {
      "name" : "ID",
      "sqlType" : "VARCHAR2",
      "maxLength" : 255,
      "assignmentMethod" : "CLIENT"
   },

   "contentColumn" :
   {
      "name" : "JSON_DOCUMENT",
      "sqlType" : "BLOB"
   },

   "versionColumn" :
   {
     "name" : "VERSION",
     "method" : "UUID"
   },

   "lastModifiedColumn" :
   {
     "name" : "LAST_MODIFIED"
   },

   "creationTimeColumn" :
   {
      "name" : "CREATED_ON"
   },

   "mediaTypeColumn" :
   {
      "name" : "YOUR_MEDIA_TYPE_COLUMN_NAME"
   },

   "readOnly" : false
}
```

Cloud Support and Identity Information

This lists the procedure to file a service request using Oracle Cloud Support.

- Obtain Tenancy Details
  When you file a service request or when you need Autonomous Database tenancy details for other reasons you can obtain these details for your instance. Tenancy details for an instance are available on the Oracle Cloud Infrastructure Console or you can obtain these details by querying the database.

Obtain Tenancy Details

When you file a service request or when you need Autonomous Database tenancy details for other reasons you can obtain these details for your instance. Tenancy details for an instance are available on the Oracle Cloud Infrastructure Console or you can obtain these details by querying the database.

If you need to file a service request use Oracle Cloud Support or contact your support representative and provide the first five of the following items (service requests do not require that you supply the outbound IP address or the availability domain):

- Database Name
- Region
- Tenancy OCID
- Database OCID
- Compartment OCID
- Outbound IP Address
- Availability Domain

If you are connected to the database you can obtain tenancy details by querying the `CLOUD_IDENTITY` column of the `V$PDBS` view.

For example, running the following:

```
SELECT cloud_identity FROM v$pdbs;

CLOUD_IDENTITY
--------------------------------------------------------------------
{
  "DATABASE_NAME" : "DBxxxxxxxxxxx",
  "REGION" : "us-phoenix-1",
  "TENANT_OCID" : "OCID1.TENANCY.REGION1..ID1",
  "DATABASE_OCID" : "OCID1.AUTONOMOUSDATABASE.OC1.IAD.ID2",
  "COMPARTMENT_OCID" : "ocid1.tenancy.region1..ID3"
  "OUTBOUND_IP_ADDRESS" :
  [
       "192.0.2.254"
  ]
  "AVAILABILITY_DOMAIN" : "SoSC:region-1-AD-1"
}
```

If your Autonomous Database instance was created before the tenancy details feature was added and has not been restarted, then this query does not return tenancy details. In this case, as a one-time operation, restart your instance and run the query again. You can restart your instance from the Oracle Cloud Infrastructure console or using the restart API.

Public IP Address Ranges for Autonomous Database

Oracle Cloud Infrastructure provides information about public IP address ranges for Autonomous Databases that are deployed in Oracle Cloud Infrastructure.

See IP Address Ranges for more information.

- Database Name
  You set the database name when you provision a database or when you rename a database.
- Region
- Tenancy OCID
- Database OCID
- Compartment OCID
- Outbound IP Address
- Availability Domain
  You can view the Availability Domain of your Autonomous Database instance.

Database Name

You set the database name when you provision a database or when you rename a database.

The Oracle Cloud Infrastructure Console shows the database name on the Autonomous Database Details page under General Information in the **Database Name** field.

Region

The Oracle Cloud Infrastructure Console shows the region on the console, in the Regions area.

Tenancy OCID

The tenancy details page shows the tenancy OCID.

See Managing the Tenancy for information on accessing the Tenancy Details page.

See Resource Identifiers for information on Oracle Cloud Identifiers.

Database OCID

The Oracle Cloud Infrastructure Console shows the database OCID on the Autonomous Database Details page under General Information in the **OCID** field.

See Resource Identifiers for information on Oracle Cloud Identifiers.

Compartment OCID

See Managing Compartments for more information on compartments.

See Resource Identifiers for information on Oracle Cloud Identifiers.

Outbound IP Address

The `OUTBOUND_IP_ADDRESS` is the outbound IP address of your Autonomous Database instance. You can use the `OUTBOUND_IP_ADDRESS` when you create a database link to another Autonomous Database instance that uses ACLs to restrict access. In this case, you need to allow the specified outbound IP address to connect to the target Autonomous Database and then create the database link.

The `OUTBOUND_IP_ADDRESS` shows the outbound IP address in the following cases:

- When your Autonomous Database instance uses a Public Endpoint.
- When your Autonomous Database instance uses a Private Endpoint and the database property `ROUTE_OUTBOUND_CONNECTIONS` is set to `''` (the default value).

> **Note:**
>
> When your Autonomous Database instance uses a Private Endpoint and you set the database property `ROUTE_OUTBOUND_CONNECTIONS` to `'PRIVATE_ENDPOINT'`, outbound connections go through the private endpoint. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

See Create Database Links from Autonomous Database to a Publicly Accessible Autonomous Database with a Wallet (mTLS) for information on creating database links.

See Configuring Network Access with Access Control Rules (ACLs) for information on configuring access control rules.

Availability Domain

You can view the Availability Domain of your Autonomous Database instance.

This is useful when you need to deploy your Application Servers when your Autonomous Database instance resides in a multi-Availability Domain region. In order to reduce latency as much as possible you would want to deploy your Application Server and database in the same Availability Domain.

See Regions and Availability Domains for more information on Availability Domains.