## Contents

### Preface

- Audience xiv
- Documentation Accessibility xiv
- Related Documents xiv
- Conventions xv

### Part I  Using Autonomous Data Warehouse

#### 1  Getting Started with Autonomous Data Warehouse

- About Autonomous Data Warehouse 1-1
- Before You Begin with Autonomous Data Warehouse 1-3
- Key Features of Autonomous Data Warehouse 1-4
- Typical Workflow for Using Autonomous Data Warehouse 1-5
- Use Oracle Machine Learning with Autonomous Data Warehouse 1-6
- Build Reports and Dashboards with Analytics in Autonomous Data Warehouse 1-7
- Use Sample Data Sets in Autonomous Data Warehouse 1-7
- Security and Authentication in Autonomous Data Warehouse 1-9
- Always Free Autonomous Database 1-9
- Preview Versions for Autonomous Database 1-12
- Oracle Database Versions and Availability by Region 1-13
- Quickstart Tutorials 1-15
  - Autonomous Database Quickstart 1-15
  - Analyzing Your Data with Autonomous Database 1-15
  - Quickstart Tutorials for Oracle Machine Learning with Autonomous Data Warehouse 1-16

#### 2  Connecting to Autonomous Data Warehouse

- About Connecting to an Autonomous Data Warehouse Instance 2-1
- Connect Autonomous Data Warehouse Using a Client Application 2-4
  - About Connecting to Autonomous Data Warehouse Using a Client Application 2-5
Prepare for Oracle Call Interface (OCI), ODBC, and JDBC OCI Connections 2-5
Prepare for JDBC Thin Connections 2-8
Using Applications with Support for Wallets 2-8
Download Client Credentials (Wallets) 2-9
Connect to Autonomous Data Warehouse Using Oracle Database Tools 2-11
  Connect with Oracle SQL Developer (18.2 or later) 2-11
  Connect with Oracle SQL Developer (earlier than Version 18.2) 2-14
  Connect with SQL*Plus 2-16
  Connect with Oracle SQLcl Cloud Connection 2-17
Connect with Built-in SQL Developer Web 2-18
  About SQL Developer Web 2-18
  Access SQL Developer Web as ADMIN 2-19
  Provide SQL Developer Web Access to Database Users 2-20
JDBC Thin Connections and Wallets 2-22
  JDBC Thin Driver Connection Prerequisites 2-22
  Using a JDBC URL Connection String with JDBC Thin Driver 2-23
  Using a JDBC Connection with 18.3 JDBC Driver 2-24
  Connecting Using JDBC Thin Driver 12.2 or Older 2-26
  JDBC Thin Connections with an HTTP Proxy 2-28
Oracle Call Interface (OCI) Connections and Wallets 2-29
Predefined Database Service Names for Autonomous Data Warehouse 2-30
Connect with Oracle Analytics Desktop 2-31
Connect with Oracle Analytics Cloud 2-31
Connect Applications to Autonomous Data Warehouse 2-31
  Connect with Microsoft .NET and Visual Studio 2-31
  Connect with JDBC Thin Driver and UCP 2-32
  Connect with Python, Node.js, and other Scripting Languages 2-32
Connection and Networking Options and Features 2-35
  Using ACLs, VCNs, and Private Endpoints with Autonomous Database 2-36
    About Network Access Options 2-36
    Overview of Restricting Access with ACLs 2-37
    Overview of Private Endpoints 2-38
  Connect with Oracle Cloud Infrastructure FastConnect 2-38
  Access Autonomous Data Warehouse with VCN Transit Routing 2-38
  Access Autonomous Data Warehouse with Service Gateway 2-39
  Enable and Disable Application Continuity 2-39
  Use Database Resident Connection Pooling with Autonomous Database 2-41
Use Database Links with Autonomous Data Warehouse 2-42
  Create Database Links from Autonomous Database to Other Databases 2-42
  Create Database Links from Other Databases to Autonomous Database 2-46
3  Loading Data with Autonomous Data Warehouse

About Data Loading 3-1
Load Data from Files in the Cloud 3-2
   Create Credentials and Copy Data into an Existing Table 3-2
   Create Credentials and Load Data Pump Dump Files into an Existing Table 3-4
Load JSON Documents with Autonomous Database 3-6
   About Loading JSON Documents 3-6
   Load a Line-Delimited JSON Document into a Collection 3-6
   Load an Array of JSON Documents into a Collection 3-8
   Monitor and Troubleshoot COPY_COLLECTION Loads 3-10
Monitor and Troubleshoot Loads 3-11
List Credentials 3-13
Delete Credentials 3-13
Import Data Using Oracle Data Pump on Autonomous Data Warehouse 3-13
   Export Your Existing Oracle Database to Import into Autonomous Data Warehouse 3-14
   Import Data Using Oracle Data Pump Version 18.3 or Later 3-15
   Import Data Using Oracle Data Pump (Versions 12.2.0.1 and Earlier) 3-17
   Access Log Files for Data Pump Import 3-19
Load Data from Local Files with SQL Developer Web 3-19
   Load Data into Existing Autonomous Database Table with SQL Developer Web 3-20
Use Oracle GoldenGate to Replicate Data to Autonomous Data Warehouse 3-24
Load Data from Local Files Using SQL*Loader 3-24
Managing DML Performance and Compression 3-24

4  Querying External Data with Autonomous Data Warehouse

Query External Data 4-1
Query External Data with Parquet or Avro Source Files 4-3
Query External Partitioned Data 4-6
Query Hybrid Partitioned Data 4-8
Query External Data Pump Dump Files 4-10
Validate External Data 4-12
Validate External Partitioned Data 4-13
Validate Hybrid Partitioned Data 4-14
View Logs for Data Validation 4-15
### Creating Dashboards, Reports, and Notebooks with Autonomous Data Warehouse

- **About Creating Dashboards, Reports, and Notebooks with Oracle Machine Learning**
  - 5-1
- **Work with Oracle Machine Learning for Data Access, Analysis, and Discovery**
  - 5-2
- **Work with Analytics and Visualization**
  - 5-4
  - Using Oracle Analytics Cloud with Autonomous Data Warehouse
    - 5-4
  - Working with Oracle Analytics Desktop in Autonomous Data Warehouse
    - 5-5

### Moving Data from Autonomous Data Warehouse to Other Oracle Databases

- **Move Data with Data Pump Export to an Autonomous Database Directory**
  - 6-1
  - Use Data Pump to Create a Dump File Set on Autonomous Data Warehouse
    - 6-2
  - Move Dump File Set from Autonomous Data Warehouse to Your Cloud Object Store
    - 6-4
- **Move Data with Data Pump Export to Object Store**
  - 6-5
  - Use Oracle Data Pump to Export Data to Object Store (Version 19.6 and Earlier)
    - 6-6
  - Use Oracle Data Pump to Export Data to Object Store (Version 19.7 or Later)
    - 6-9
- **Move Selective Data to Object Store Using DBMS_CLOUD.EXPORT_DATA**
  - 6-12
- **Download Dump Files, Run Data Pump Import, and Clean Up Object Store**
  - 6-14
- **Access Log Files for Data Pump Export**
  - 6-15

### Developing RESTful Services in Autonomous Database

- **About Oracle REST Data Services in Autonomous Database**
  - 7-1
- **Access RESTful Services and SODA for REST**
  - 7-2
- **Develop with Oracle REST Data Services on Autonomous Database**
  - 7-3
- **Use SODA for REST with Autonomous Database**
  - 7-4
- **About Customer Managed Oracle REST Data Services on Autonomous Database**
  - 7-6

### Creating Applications with Oracle Application Express in Autonomous Database

- **About Oracle Application Express**
  - 8-1
- **Access Oracle Application Express Administration Services**
  - 8-2
- **Create Oracle Application Express Workspaces in Autonomous Data Warehouse**
  - 8-4
- **Access Oracle Application Express App Builder**
  - 8-5
- **Create Oracle Application Express Developer Accounts**
  - 8-5
- **Use JSON Data with Oracle Application Express**
  - 8-6
<table>
<thead>
<tr>
<th>Page</th>
<th>Section Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-6</td>
<td>Create A View with JSON Data Guide</td>
<td></td>
</tr>
<tr>
<td>8-7</td>
<td>Create A View with the json_table Function</td>
<td></td>
</tr>
<tr>
<td>8-7</td>
<td>Use Web Services with Oracle Application Express</td>
<td></td>
</tr>
<tr>
<td>8-9</td>
<td>Send Email from Oracle Application Express</td>
<td></td>
</tr>
<tr>
<td>8-10</td>
<td>Restrictions and Limitations for Oracle Application Express with Autonomous Data Warehouse</td>
<td></td>
</tr>
<tr>
<td>9-1</td>
<td>Creating and Managing Directories on Autonomous Database</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Create Directory in Autonomous Database</td>
<td>9-1</td>
</tr>
<tr>
<td></td>
<td>Drop Directory in Autonomous Database</td>
<td>9-2</td>
</tr>
<tr>
<td></td>
<td>List Contents of Directory in Autonomous Database</td>
<td>9-3</td>
</tr>
<tr>
<td></td>
<td>Copy Files Between Object Store and a Directory in Autonomous Database</td>
<td>9-3</td>
</tr>
<tr>
<td>10-1</td>
<td>Sending Mail with Email Delivery on Autonomous Database</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Send Mail Using Email Delivery on Autonomous Database</td>
<td>10-1</td>
</tr>
<tr>
<td></td>
<td>SMTP Send Mail Sample Code</td>
<td>10-2</td>
</tr>
<tr>
<td>11-1</td>
<td>Use Oracle Extensions for IDEs to Develop Applications</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use Oracle Cloud Infrastructure Toolkit for Eclipse</td>
<td>11-1</td>
</tr>
<tr>
<td></td>
<td>Use Oracle Developer Tools for Visual Studio</td>
<td>11-2</td>
</tr>
<tr>
<td></td>
<td>Use Oracle Developer Tools for VS Code</td>
<td>11-3</td>
</tr>
<tr>
<td>12-1</td>
<td>Using JSON Documents with Autonomous Database</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Work with Simple Oracle Document Access (SODA) in Autonomous Database</td>
<td>12-1</td>
</tr>
<tr>
<td></td>
<td>Work with JSON Documents Using SQL and PL/SQL APIs on Autonomous Database</td>
<td>12-3</td>
</tr>
<tr>
<td></td>
<td>Load JSON Documents with Autonomous Database</td>
<td>12-3</td>
</tr>
<tr>
<td>13-1</td>
<td>Starting, Stopping, and Scaling Autonomous Data Warehouse</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Provision Autonomous Data Warehouse</td>
<td>13-1</td>
</tr>
<tr>
<td></td>
<td>Start Autonomous Data Warehouse</td>
<td>13-4</td>
</tr>
<tr>
<td></td>
<td>Stop Autonomous Data Warehouse</td>
<td>13-4</td>
</tr>
<tr>
<td></td>
<td>Restart Autonomous Data Warehouse</td>
<td>13-5</td>
</tr>
<tr>
<td></td>
<td>Terminate Autonomous Data Warehouse</td>
<td>13-5</td>
</tr>
<tr>
<td></td>
<td>Add CPU or Storage Resources or Enable Auto Scaling</td>
<td>13-6</td>
</tr>
</tbody>
</table>
14 Managing Users on Autonomous Data Warehouse

Create Users with Autonomous Data Warehouse 14-1
Remove Users with Autonomous Data Warehouse 14-3
Manage the Administrator Account on Autonomous Data Warehouse 14-3
  Change the Administrator Password in Autonomous Data Warehouse 14-3
  Unlock the Administrator Account in Autonomous Data Warehouse 14-4
Manage User Privileges with Autonomous Data Warehouse 14-5
Create and Update User Accounts for Oracle Machine Learning 14-6
  Create User 14-6
  Add Existing Database User Account to Oracle Machine Learning 14-8
Use Microsoft Active Directory with Autonomous Database 14-9
  Configure CMU with Microsoft Active Directory on Autonomous Database 14-9
  Add Microsoft Active Directory Roles on Autonomous Database 14-13
  Add Microsoft Active Directory Users on Autonomous Database 14-14
  Connect to Autonomous Database with Active Directory User Credentials 14-15
  Tools Restrictions with Active Directory on Autonomous Database 14-16
  Verify Active Directory User Connection Information with Autonomous Database 14-16
  Remove Active Directory Users and Roles on Autonomous Database 14-17
  Disable Active Directory Access on Autonomous Database 14-17

15 Managing and Monitoring Performance of Autonomous Data Warehouse

Monitor the Performance of Autonomous Data Warehouse 15-1
  Monitor Activity and Utilization 15-2
  Console Overview 15-3
  Console Activity 15-4
  Monitor SQL Statements 15-8
Managing Concurrency and Priorities on Autonomous Data Warehouse 15-10
Manage CPU/IO Shares on Autonomous Data Warehouse 15-12
Manage Runaway SQL Statements on Autonomous Data Warehouse 15-14
Manage Optimizer Statistics on Autonomous Data Warehouse 15-15
Manage Automatic Indexing on Autonomous Database 15-16
Monitor Autonomous Data Warehouse with Performance Hub 15-17
  Monitor the Performance of Autonomous Data Warehouse with Oracle Management Cloud 15-19
Monitor Performance with Autonomous Database Metrics 15-19
  View Metrics for an Autonomous Database Instance 15-20
16 **Back up and Restoring Autonomous Data Warehouse**

- **About Backup and Recovery on Autonomous Data Warehouse**
- **Restore and Recover your Autonomous Data Warehouse Database**
- **Manual Backups on Autonomous Data Warehouse**
  - Configure Manual Backups on Autonomous Data Warehouse
  - Perform Manual Backups on Autonomous Data Warehouse

17 **Cloning, Moving, or Upgrading an Autonomous Data Warehouse Database**

- **Clone an Autonomous Data Warehouse Instance**
- **Clone Autonomous Data Warehouse from a Backup**
- **Notes for Cloning Autonomous Database**
  - Resource Management Rules and Performance Data for a Cloned Database
  - Optimizer Statistics for a Cloned Database
- **Move an Autonomous Data Warehouse Database to a Different Compartment**
- **Upgrade an Autonomous Database Instance to Oracle Database 19c**
  - Manually Upgrade to Oracle Database 19c
  - Manually Upgrade by Performing a Full Clone
  - Manually Upgrade by Cloning from a Backup
  - Automatically Upgrade to Oracle Database 19c

18 **Wallet Security, Data Safe, and Oracle Database Vault**

- **Rotate Wallets for Autonomous Database**
- **Update License Type on Autonomous Database**
- **Update Always Free Instance to Paid with Autonomous Database**
- **Safeguard Your Data with Data Safe on Autonomous Database**
- **Use Oracle Database Vault with Autonomous Database**
  - Oracle Database Vault Users and Roles on Autonomous Database
  - Enable Oracle Database Vault on Autonomous Database
  - Disable Oracle Database Vault on Autonomous Database
  - Disable User Management with Oracle Database Vault on Autonomous Database
  - Enable User Management with Oracle Database Vault on Autonomous Database
19 Configuring Network Access with Access Control Rules (ACLs) and Private Endpoints

Configure an Access Control List with Autonomous Database 19-1
Configure Private Endpoints with Autonomous Database 19-4
Private Endpoint Configuration Examples on Autonomous Database 19-7

20 Maintenance Windows, Work Requests and Managing Events and Notifications

About Events and Notifications 20-1
About Work Requests 20-1
About Autonomous Database Maintenance Windows 20-3

21 Using a Standby Database with Autonomous Database

About Autonomous Data Guard 21-1
   Autonomous Data Guard Features 21-1
   Autonomous Database Failover with Autonomous Data Guard 21-2
   Autonomous Data Guard Operations 21-3
   Autonomous Database Standby Database States 21-3
   Autonomous Data Guard Notes 21-4
Enable Autonomous Data Guard 21-4
Disable Autonomous Data Guard 21-6
Switchover Autonomous Data Guard 21-7
Manual Failover with Autonomous Data Guard 21-8

Part III Appendixes

A Autonomous Database Supplied Package Reference

Summary of DBMS_CLOUD Subprograms A-1
   COPY_COLLECTION Procedure A-2
   COPY_DATA Procedure A-3
   COPY_DATA Procedure for Parquet or Avro Files A-5
   CREATE_CREDENTIAL Procedure A-6
   CREATE_CREDENTIAL Procedure (OCI Signing Key Credentials) A-7
   CREATE_EXTERNAL_TABLE Procedure A-8
   CREATE_EXTERNAL_TABLE Procedure for Parquet or Avro Files A-10
   CREATEEXTERNAL_PART_TABLE Procedure A-11
   CREATE_HYBRID_PART_TABLE Procedure A-13
Autonomous Data Warehouse for Experienced Oracle Database Users

About the Autonomous Data Warehouse Database B-1
Autonomous Data Warehouse Oracle Database 19c Features B-3
Restrictions for Database Initialization Parameters B-3
Restrictions for SQL Commands B-4
Restrictions for Data Types B-7
Managing Partitions, Indexes, and Materialized Views B-9
Restrictions and Notes for Database PL/SQL Packages B-9
Restrictions for Database Features B-11
  Restrictions for Oracle XML DB B-11
  Restrictions for Oracle Text B-12
  Restrictions for Oracle Spatial and Graph B-13
  Restrictions for Oracle Application Express B-14
  Restrictions for Oracle Flashback B-14
  Restrictions for Fast Application Notification (FAN) B-15
List of Restricted and Removed Oracle Features B-15

Migrating Amazon Redshift to Autonomous Data Warehouse

Autonomous Data Warehouse Redshift Migration Overview C-1
Connect to Amazon Redshift C-2
Connect to Autonomous Data Warehouse C-4
Start the Cloud Migration Wizard C-5
Review and Finish the Amazon Redshift Migration C-11
Use Generated Amazon Redshift Migration Scripts C-12
Perform Post Migration Tasks C-14

Sample Star Schema Benchmark (SSB) Queries and Analytic Views

Star Schema Benchmark Queries D-1
Star Schema Benchmark Analytic Views D-4

SODA Collection Metadata on Autonomous Database

SODA Default Collection Metadata on Autonomous Database E-1
F Obtain Tenancy Details to File a Service Request
Preface

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

Audience

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

This document is also intended for developers and end users who want to load and query data in Oracle Autonomous Data Warehouse.

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 for your Autonomous Data Warehouse database is either Oracle Database 18c or Oracle Database 19c.

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

Oracle Database 19c

If you have Oracle Database 18c, then many concepts and features of this service are further documented here:

Oracle Database 18c

See Oracle Database Versions and Availability by Region for details on regions and Oracle Database availability.

For additional information, see these Oracle resources:

- Getting Started with Oracle Cloud
Conventions

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td><em>italic</em></td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td>monospace</td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
Part I
Using Autonomous Data Warehouse

This part provides information on using Autonomous Data Warehouse.

Topics

• Getting Started with Autonomous Data Warehouse
• Connecting to Autonomous Data Warehouse
• Loading Data with Autonomous Data Warehouse
• Querying External Data with Autonomous Data Warehouse
• Creating Dashboards, Reports, and Notebooks with Autonomous Data Warehouse
• Moving Data from Autonomous Data Warehouse to Other Oracle Databases
• Creating Applications with Oracle Application Express in Autonomous Database
• Developing RESTful Services in Autonomous Database
• Creating and Managing Directories on Autonomous Database
• Sending Mail with Email Delivery on Autonomous Database
• Use Oracle Extensions for IDEs to Develop Applications
• Using JSON Documents with Autonomous Database
Getting Started with Autonomous Data Warehouse

Provides an overview of the service and describes how to get started with Autonomous Data Warehouse.

Topics

- About Autonomous Data Warehouse
- Before You Begin with Autonomous Data Warehouse
- Key Features of Autonomous Data Warehouse
- Typical Workflow for Using Autonomous Data Warehouse
- Use Oracle Machine Learning with Autonomous Data Warehouse
- Build Reports and Dashboards with Analytics in Autonomous Data Warehouse
- Use Sample Data Sets in Autonomous Data Warehouse
- Security and Authentication in Autonomous Data Warehouse
- Always Free Autonomous Database
- Preview Versions for Autonomous Database
- Oracle Database Versions and Availability by Region
- Quickstart Tutorials

About Autonomous Data Warehouse

Autonomous Data Warehouse provides an easy-to-use, fully autonomous data warehouse that scales elastically, delivers fast query performance and requires no database administration. It 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.

As a service, Autonomous Data Warehouse does not require database administration. With Autonomous Data Warehouse you do not need to configure or manage any hardware, or install any software. Autonomous Data Warehouse handles creating the data warehouse, backing up the database, patching and upgrading the database, and growing or shrinking the database.

Additionally, Autonomous Data Warehouse does not require any tuning. Autonomous Data Warehouse is designed as a "load and go" service: you start the service, define tables, load data, and then run queries. When you use Autonomous Data Warehouse, no tuning is necessary. You do not need to consider any details about parallelism, partitioning, indexing, or compression. The service automatically configures the database for high-performance queries.
Autonomous Data Warehouse is built upon Oracle Database, so that the applications and tools that support Oracle Database also support Autonomous Data Warehouse. These tools and applications connect to Autonomous Data Warehouse 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 Data Warehouse connections.

Autonomous Data Warehouse is a completely elastic service. When you get started with Autonomous Data Warehouse, simply specify the number of OCPUs and the storage capacity in TB’s for the data warehouse. At any time, you can scale, increase or decrease, either the OCPUs or the storage capacity. When you make resource changes for your Autonomous Data Warehouse, the data warehouse resources automatically shrink or grow, without requiring any downtime or service interruptions.

Autonomous Data Warehouse includes a cloud-based service console for managing the service (for tasks such as stopping, starting, or scaling the service), and monitoring the service (for tasks such as viewing the recent levels of activity on the data warehouse).

Autonomous Data Warehouse also includes the following:

- Oracle Application Express (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.
- Oracle SQL Developer Web: a browser-based interface of Oracle SQL Developer.
- Oracle Machine Learning: a cloud-based notebook application which provides simple querying, data-visualization, and collaboration capabilities. The notebook is designed to be used alongside other business intelligence applications.

You can use Autonomous Data Warehouse 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 Data Warehouse architecture.
Before You Begin with Autonomous Data Warehouse

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

See *Getting Started with Oracle Cloud*.

Before you create an Autonomous Data Warehouse instance:

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

- (Optional) if you want to leverage an object store for data loading you need your object store credentials to use with Oracle Autonomous Data Warehouse, 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*.

- (Optional) Create a bucket for cloud storage manual backups:
If you want to manually back up Autonomous Data Warehouse to cloud storage you must associate an instance with a cloud storage backup location. For information on defining your Oracle Cloud Infrastructure Object Storage and creating the bucket for manual backups, see Configure Manual Backups on Autonomous Data Warehouse.

After you create an Autonomous Data Warehouse instance:

- (Optional) Reset the administrator password:
  When you create an Autonomous Data Warehouse instance 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 Data Warehouse.

- (Optional) Add Oracle Machine Learning users. See: Create User.

- (Optional) Download Oracle Analytics Desktop and add a connection to Autonomous Data Warehouse. See Work with Analytics and Visualization.

Key Features of Autonomous Data Warehouse

This section describes key features of Autonomous Data Warehouse, an affordable, feature-rich service in the cloud.

Key Features

- **Managed**: Oracle simplifies end-to-end management of the data warehouse:
  - Provisioning new databases
  - Growing or shrinking storage and compute resources
  - Patching and upgrades
  - Backup and recovery

- **Fully Tuned**: “Load and go”:
  - Define tables, load data, run queries
  - Provides good performance out of the box
  - Run your queries using any business analytics tool or cloud service
  - Built-in SQL worksheet and notebook also included

- **Fully elastic scaling**: Scale compute and storage independently to fit your data warehouse workload with no downtime:
  - Size the Autonomous Data Warehouse to the exact compute and storage required
  - Scale the Autonomous Data Warehouse 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 automatically when the workload requires it:
  - Specify the number of OCPUs for Autonomous Data Warehouse workloads.
  - Auto scaling allows the database to use up to three times more CPU and IO resources depending on workload requirements.
Auto scaling is enabled by default when you create an Autonomous Database instance or you can use **Scale Up/Down** on the Oracle Cloud Infrastructure console to enable or disable auto scaling.

### Autonomous Data Warehouse 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, Oracle Integration Service, and others

### High-performance queries and concurrent workloads:
Optimized query performance with preconfigured resource profiles for different types of users.

### Oracle SQL:
Autonomous Data Warehouse 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 Amazon AWS Redshift, SQL Server, and other databases.

## Simple Cloud-based Data Loading

Autonomous Data Warehouse provides:
- Fast, scalable data-loading from Oracle Object Store, Azure Blob Storage, AWS S3, or on-premise data sources.

## SQL Developer Autonomous Data Warehouse Support

Using Autonomous Data Warehouse with SQL Developer you can do the following:
- Connect to Autonomous Data Warehouse
- Create tables in Autonomous Data Warehouse
- Load data into Autonomous Data Warehouse
- Copy tables to Autonomous Data Warehouse
- Transfer a schema to Autonomous Data Warehouse

## Business Intelligence Tools Support

Autonomous Data Warehouse 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

### Typical Workflow for Using Autonomous Data Warehouse

Describes the steps to start using Autonomous Data Warehouse.
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create and log in to your cloud account</strong></td>
<td>Provide your information and sign up for an Oracle Cloud Service.</td>
<td>Getting Started with Oracle Cloud</td>
</tr>
<tr>
<td><strong>Provision Autonomous Data Warehouse</strong></td>
<td>Create an Autonomous Data Warehouse instance.</td>
<td>Autonomous Database Quickstart</td>
</tr>
<tr>
<td><strong>Start the data warehouse</strong></td>
<td>Start and verify that an Autonomous Data Warehouse instance is up and running.</td>
<td>Start Autonomous Data Warehouse</td>
</tr>
<tr>
<td><strong>Create database users and obtain security credentials</strong></td>
<td>Create accounts for your users and assign them appropriate privileges. Assign the necessary Autonomous Data Warehouse roles.</td>
<td>Create Users with Autonomous Data Warehouse</td>
</tr>
<tr>
<td><strong>Connect to your data warehouse using SQL Developer</strong></td>
<td>Download SQL Developer or another database client tool. Obtain security credentials and connect to your Autonomous Data Warehouse.</td>
<td>Connect to Autonomous Data Warehouse Using Oracle Database Tools</td>
</tr>
<tr>
<td><strong>Connect to your data warehouse using a database client</strong></td>
<td>Download a database client. Obtain security credentials and connect to your Autonomous Data Warehouse instance.</td>
<td>Connecting to Autonomous Data Warehouse</td>
</tr>
<tr>
<td><strong>Scale the data warehouse</strong></td>
<td>Use the Oracle Cloud Infrastructure Console to scale an instance to add or remove CPU or storage resources.</td>
<td>Starting, Stopping, and Scaling Autonomous Data Warehouse</td>
</tr>
<tr>
<td><strong>Load data into the data warehouse</strong></td>
<td>Use the available tools to load data into the data warehouse.</td>
<td>Loading Data with Autonomous Data Warehouse</td>
</tr>
<tr>
<td><strong>Migrate data from Amazon Redshift</strong></td>
<td>Migrate data from an existing Amazon Redshift data warehouse.</td>
<td>Migrating Amazon Redshift to Autonomous Data Warehouse</td>
</tr>
<tr>
<td><strong>Monitor the data warehouse</strong></td>
<td>Check on the health and performance of the data warehouse.</td>
<td>Managing and Monitoring Performance of Autonomous Data Warehouse</td>
</tr>
<tr>
<td><strong>Manage the data warehouse</strong></td>
<td>Check on the day-to-day operation of your service, monitor performance, and review important notifications.</td>
<td>All aspects of the database lifecycle are simplified by the service, including patching and upgrading the database. You can obtain service information including the service and database version from the service console. Backing Up and Restoring Autonomous Data Warehouse</td>
</tr>
<tr>
<td><strong>Clone the data warehouse</strong></td>
<td>Use the Oracle Cloud Infrastructure console to clone a database. You can choose to clone either the full database or only the database metadata.</td>
<td>Cloning, Moving, or Upgrading an Autonomous Data Warehouse Database</td>
</tr>
</tbody>
</table>

**Use Oracle Machine Learning with Autonomous Data Warehouse**

Oracle Machine Learning provides a notebook style application designed for advanced SQL users and 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:**
• 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 with Autonomous Data Warehouse you need to add or create Oracle Machine Learning users and access Oracle Machine Learning:

• **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](#) for details on accessing OML User Management.

• **OML Application** – application **users** access Oracle Machine Learning to create, view, and share notebooks for data analytics, data visualization, and other Oracle Machine Learning tasks. Users access Oracle Machine Learning 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 for Data Access, Analysis, and Discovery](#) for details on using the OML application.

## Build Reports and Dashboards with Analytics in Autonomous Data Warehouse

### Working with Oracle Analytics Cloud

You can use Oracle Analytics Cloud with Autonomous Data Warehouse. Use Oracle Analytics Cloud to select interactive visualizations and automatically create advanced calculations to reveal the insights in your data.

For more information, see [Using Oracle Analytics Cloud with Autonomous Data Warehouse](#).

### Working with Oracle Analytics Desktop

You can use Oracle Analytics Desktop with Autonomous Data Warehouse. Just connect to Autonomous Data Warehouse, 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 [Working with Oracle Analytics Desktop in Autonomous Data Warehouse](#).

## Use Sample Data Sets in Autonomous Data Warehouse

For users who want to start using the service without creating their own tables, Autonomous Data Warehouse 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.
Note:

Both SH and SSB are provided as schema-only users, so you cannot unlock or drop those users or set a password. The storage of the sample data sets does not count towards your database 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:

```sql
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 in the Autonomous Data Warehouse contains 1 TB of data. 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 OCPUs to test the performance of different Autonomous Data Warehouse configurations.

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 Data Warehouse.
Security and Authentication in Autonomous Data Warehouse

Autonomous Data Warehouse stores all data in encrypted format in the Oracle Database. Only authenticated users and applications can access the data when they connect to the database.

All connections to Autonomous Data Warehouse use certificate-based authentication and Secure Sockets Layer (SSL). This ensures that there is no unauthorized access to Autonomous Data Warehouse and that communications between the client and server are fully encrypted and cannot be intercepted or altered.

Certificate based authentication 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 Data Warehouse 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 database service in the Autonomous Data Warehouse. For more information on connections to Autonomous Data Warehouse see About Connecting to an Autonomous Data Warehouse Instance.

You do not need to do any manual configuration to encrypt your data and the connections to your database. These are implemented by Autonomous Data Warehouse.

Autonomous Data Warehouse 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 with Autonomous Data Warehouse.

You can further restrict connections by specifying a network Access Control List (ACL). By specifying a network ACL a specific Autonomous Data Warehouse database only accepts connections from addresses on the ACL and rejects all other client connections. See Overview of Restricting Access with ACLs for more information.

When you provision or clone an Autonomous Database, you can configure the network access so that the database uses a private endpoint. If your organization has strict security mandates that do not allow you to have a public endpoint for your database, this provides you with the necessary private endpoint. When you use private access, your database is only accessible through the IP address of the associated private endpoint. Additionally, this allows you to configure your access so that the traffic does not use public subnets and allows you to keep all traffic to and from your Autonomous Database off of the public internet within one of your tenancy's virtual cloud networks (VCNs). See Overview of Private Endpoints for more information.

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.

**Resource Restrictions for Always Free Autonomous Database**

- Maximum of 1 OCPU per database
- Maximum of approximately 20 GB Exadata storage per database (you may see more than this)
- Maximum of 20 simultaneous database sessions
- Maximum of 2 Always Free databases per Oracle Cloud Infrastructure tenancy. There is no restriction on the workload type for Always Free databases in a tenancy. If you create 2 Always Free databases both can have the same workload type or they can have different workload types. Both Data Warehouse and Transaction Processing workload types are supported.

**Notes:**

- Always Free Autonomous Databases cannot be scaled manually or automatically beyond the fixed resource restrictions described above.
- The Maximum of 20 simultaneous database sessions limit for Always Free and 1 OCPU per database allows you to work with Autonomous Databases; 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 with private access using a private endpoint. See [Configure Private Endpoints with Autonomous Database](#) for information on private access and private endpoints.

**Oracle Database Version and Oracle Application Express Release**

Always Free Autonomous Database uses the following:

- The available Database versions for Always Free Autonomous Database are: Oracle Database 18c or Oracle Database 19c.

If you are currently on Oracle Database 18c, you can wait for your Always Free instance to be automatically upgraded to Oracle Database 19c. After notifying you of the upgrade date, the automatic upgrade will be performed during one of your database's upcoming maintenance windows. You can see the maintenance window for your database on the Autonomous Database Details page. The automatic upgrade is a minimal downtime operation and requires up to ten (10) minutes of downtime during the maintenance window.

- The Oracle Application Express Release for Always Free Autonomous Database is: [Oracle Application Express Release 19.2](#)

**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. See Regions and Availability for further information.

- You can create Always Free Autonomous Databases in your home data region. When you sign up for Oracle Cloud Infrastructure, Oracle creates a tenancy in a data region that you select, which becomes your home 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 full backups to your Oracle Cloud Infrastructure object storage.
- Always Free Autonomous Databases do not support restoring from full backups.

Preview Versions Not Available in Always Free Autonomous Database

Oracle periodically provides a preview version of Autonomous Database that includes new Autonomous Database features that you can use before these features are made available in existing Autonomous Databases. Preview versions of Autonomous Database are not available with Always Free Autonomous Databases. See Preview Versions for Autonomous Database for information on preview versions.

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. Successfully making a SQL*Net or HTTPS connection 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.

Note:

When you start an Always Free Autonomous Database instance from 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.
Upgrading Always Free Autonomous Databases to Paid Instances

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

If your Oracle Cloud Infrastructure account is in a trial period or has paying status, you can directly upgrade an Always Free database to a paid instance as follows:

1. Sign in to your Oracle Cloud Account at cloud.oracle.com.
2. From the Oracle Cloud Infrastructure left navigation list, expand Services, and click Autonomous Data Warehouse.
   1. Select the Always Free instance by clicking the link under Name.
   2. On the Details page, from the More Actions drop-down list select Update Instance to Paid.
3. Confirm and proceed with the upgrade.

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 a paid instances until the account is first upgraded to paying status. See Upgrade Your Free Oracle Cloud Promotion for more information.

Preview Versions 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.

Preview Version Overview

When an Autonomous Database preview version is available you can provision an Autonomous Database to try new features and to test your applications before the general availability of these features. Preview versions of Autonomous Database are typically available for a limited time.

Do not use preview versions for production databases or for databases that need to persist beyond the limited preview period. Note that preview version databases and their associated resources, including backups, are terminated automatically at the conclusion of the preview period. Oracle will notify you prior to the conclusion of the preview period regarding the end date of the preview. Databases that you provision or clone with a preview version display the end date of the preview period at the top of the Autonomous Database details page in Oracle Cloud Infrastructure Console.

Create a Preview Version with Autonomous Database

You create a preview version of Autonomous Database either by provisioning a new database or by cloning an existing database:
• To provision a new preview version of Autonomous Database follow the steps to provision a database and select **Enable Preview Mode** and accept the preview termination notice. See **Provision Autonomous Data Warehouse** for details on provisioning Autonomous Databases.

• To clone an Autonomous Database to a preview version follow the steps to clone a database and select **Enable Preview Mode** and accept the preview termination notice. See **Clone an Autonomous Data Warehouse Instance** for details on cloning Autonomous Databases.

**Note:**

Any existing Autonomous Database, including those provisioned with preview version, can be cloned using the preview version of Autonomous Database. Preview version databases cannot be cloned to a non-preview version of Autonomous Database.

Oracle Database Versions and Availability by Region

Describes Oracle Database version availability with Autonomous Database and shows the versions of Oracle Database available in Autonomous Database on Shared Exadata Infrastructure by region.

**Oracle Database Versions**

Depending on the region where you provision or clone your database, Autonomous Database supports one or more Oracle Database versions. The time for Oracle Database version availability depends on the region (some regions do not support multiple Oracle Database versions).

When multiple database versions are available, you choose an Oracle Database version when you provision or clone a database:

• See **Provision Autonomous Data Warehouse** for information on selecting an Oracle Database version when you provision a database.

• See **Clone an Autonomous Data Warehouse Instance** for information on selecting an Oracle Database version when you clone a database instance.

• See **Clone Autonomous Data Warehouse from a Backup** for information on selecting an Oracle Database version when you clone a database from a backup.
### Asia Pacific: APAC

<table>
<thead>
<tr>
<th>Region</th>
<th>Oracle Database Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Australia East (Sydney)</td>
<td>Oracle Database 18c, Oracle Database 19c</td>
</tr>
<tr>
<td>Australia Southeast (Melbourne)</td>
<td>Oracle Database 19c</td>
</tr>
<tr>
<td>India South (Hyderabad)</td>
<td>Oracle Database 19c</td>
</tr>
<tr>
<td>India West (Mumbai)</td>
<td>Oracle Database 18c, Oracle Database 19c</td>
</tr>
<tr>
<td>Japan Central (Osaka)</td>
<td>Oracle Database 19c</td>
</tr>
<tr>
<td>Japan East (Tokyo)</td>
<td>Oracle Database 18c, Oracle Database 19c</td>
</tr>
<tr>
<td>South Korea Central (Seoul)</td>
<td>Oracle Database 18c, Oracle Database 19c</td>
</tr>
</tbody>
</table>

### Europe, Middle East, and Africa: EMEA

<table>
<thead>
<tr>
<th>Region</th>
<th>Oracle Database Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Germany Central (Frankfurt)</td>
<td>Oracle Database 18c, Oracle Database 19c</td>
</tr>
<tr>
<td>Netherlands Northwest (Amsterdam)</td>
<td>Oracle Database 19c</td>
</tr>
<tr>
<td>Saudi Arabia West (Jeddah)</td>
<td>Oracle Database 19c</td>
</tr>
<tr>
<td>Switzerland North (Zurich)</td>
<td>Oracle Database 18c, Oracle Database 19c</td>
</tr>
<tr>
<td>UK Gov South (London)</td>
<td>Oracle Database 18c</td>
</tr>
<tr>
<td>UK South (London)</td>
<td>Oracle Database 18c, Oracle Database 19c</td>
</tr>
</tbody>
</table>

### Latin America Division: LAD

<table>
<thead>
<tr>
<th>Region</th>
<th>Oracle Database Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brazil East (Sao Paulo)</td>
<td>Oracle Database 18c, Oracle Database 19c</td>
</tr>
</tbody>
</table>

### North America

<table>
<thead>
<tr>
<th>Region</th>
<th>Oracle Database Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canada Southeast (Montreal)</td>
<td>Oracle Database 18c, Oracle Database 19c</td>
</tr>
<tr>
<td>Canada Southeast (Toronto)</td>
<td>Oracle Database 18c, Oracle Database 19c</td>
</tr>
<tr>
<td>US East (Ashburn)</td>
<td>Oracle Database 18c, Oracle Database 19c</td>
</tr>
<tr>
<td>US West (Phoenix)</td>
<td>Oracle Database 18c, Oracle Database 19c</td>
</tr>
</tbody>
</table>

---

Note:

Always Free Autonomous Database only supports a single Oracle Database version. See Always Free Autonomous Database for more information.
Quickstart Tutorials

Provides links to the Autonomous Data Warehouse quickstart tutorials.

Autonomous Database Quickstart

Learn about Autonomous Database on Shared Infrastructure and learn how to create an instance in just a few clicks. Then load data into your database, query it, and visualize it.

Workshop 1: Autonomous Database Quickstart

- Login to Oracle Cloud
- Getting Started
- Working with Free Sample Data Sets
- Loading Data
- Querying External Data
- Visualizing Your Data

Analyzing Your Data with Autonomous Database

Connect using secure wallets and monitor your Autonomous Database instances. Use Oracle Analytics Desktop (OAD) to visualize data in Autonomous Database. Use Oracle Machine Learning (OML) to try your hand at predictive analytics.

Workshop 2: Analyzing your data with Autonomous Database

- Login to Oracle Cloud
- Getting Started
- Managing and Monitoring
- Working with Free Sample Data Sets
- Loading Data
- Querying External Data
- Visualizing Your Data
- Using Wallets for Secure Connections
- Scaling and Performance
- Getting Started with Machine Learning Notebooks
- Building a Simple Machine Learning Algorithm
Quickstart Tutorials for Oracle Machine Learning with Autonomous Data Warehouse

These tutorials show you how to use Oracle Machine Learning with Autonomous Data Warehouse.

- Quickstart Tutorial – Creating Projects and Workspaces in Oracle Machine Learning
- Quickstart Tutorial – Creating and Running Notebooks in Oracle Machine Learning
- Quickstart Tutorial – Collaborating in Oracle Machine Learning
- Quickstart Tutorial – Creating a SQL Script in Oracle Machine Learning
- Quickstart Tutorial – Running SQL Statements in Oracle Machine Learning
Connecting to Autonomous Data Warehouse

Describes methods to securely connect to Autonomous Data Warehouse.

Topics

- About Connecting to an Autonomous Data Warehouse Instance
- Connect Autonomous Data Warehouse Using a Client Application
- Download Client Credentials (Wallets)
- Connect to Autonomous Data Warehouse Using Oracle Database Tools
- Connect with Built-in SQL Developer Web
- JDBC Thin Connections and Wallets
- Oracle Call Interface (OCI) Connections and Wallets
- Predefined Database Service Names for Autonomous Data Warehouse
- Connect with Oracle Analytics Desktop
- Connect with Oracle Analytics Cloud
- Connect Applications to Autonomous Data Warehouse
- Connection and Networking Options and Features
- Use Database Links with Autonomous Data Warehouse

About Connecting to an Autonomous Data Warehouse Instance

Applications and tools connect to an Autonomous Data Warehouse instance 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 support a variety of connection types to the Autonomous Data Warehouse, including:

- Oracle Call Interface (OCI), which is used by many applications written in C language. Examples include Oracle utilities such as Oracle SQL*Plus, SQL*Loader, and Oracle Data Pump.
- ODBC drivers, which can be used by applications running on Microsoft Windows, are layered over Oracle Call Interface (OCI).
• 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.

• JDBC Thin Driver, also for Java applications, is a pure Java driver. Oracle SQL Developer supports JDBC Thin Driver connections.

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

Secure Connections to Autonomous Data Warehouse

Connections to Autonomous Data Warehouse are made over the public Internet, and all applications use a secure connection. If you are familiar with using an Oracle Database within your own data center, you may not have previously used these secure connections; next we describe the basic concepts of secure database connections.

Many applications provide support for more than one connection type, but each type of connection to Autonomous Data Warehouse uses certificate authentication and Secure Sockets Layer (SSL). This ensures that there is no unauthorized access to the Autonomous Data Warehouse and that communications between the client and server are fully encrypted and cannot be intercepted or altered.

Certification authentication 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 Data Warehouse 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 database service in the Autonomous Data Warehouse. All communications between the client and the server are encrypted.

The following figure shows client secure connections to Autonomous Data Warehouse.
Connecting to Autonomous Data Warehouse 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.

Autonomous Data Warehouse is accessed using the public internet. To access the Autonomous Data Warehouse 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 Data Warehouse 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:

```sql
my_adwc_high = (description = (
  address=(protocol=tcps)
  (port=1522)
  (host=adwc.example.oraclecloud.com))

(connect_data=(service_name=example_high.adwc.oraclecloud.com))
```
Your firewall must allow access to servers within the .oraclecloud.com domain using port 1522. To connect to Autonomous Data Warehouse, 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 is a feature that enables the replay, in a non-disruptive and rapid manner, of a request against the database after a recoverable error that makes the database session unavailable so an outage appears to the user as no more than a delayed execution of the request. With Application Continuity, Autonomous Data Warehouse, the Oracle drivers, and the Oracle connection pools all collaborate to mask many outages in a safe and reliable way.

You can change the failover type on Autonomous Data Warehouse using the DBMS_CLOUD_ADMIN procedures to enable or to disable Application Continuity. New sessions use the new failover type from the time when you modify the current value.

Note:
By default Application Continuity is disabled.

See Overview of Application Continuity for more information on Application Continuity.

See Enable and Disable Application Continuity for information on enabling and disabling Application Continuity in Autonomous Data Warehouse.

Connect Autonomous Data Warehouse Using a Client Application

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

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

Topics
• About Connecting to Autonomous Data Warehouse Using a Client Application
• Prepare for Oracle Call Interface (OCI), ODBC, and JDBC OCI Connections
• Prepare for JDBC Thin Connections
• Using Applications with Support for Wallets
About Connecting to Autonomous Data Warehouse Using a Client Application

Applications can connect to Autonomous Data Warehouse 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 Data Warehouse using a client application:

1. Determine what connection type your application uses, (for example OCI, ODBC, JDBC Thin, and so on).
2. Prepare your client computer for the type of connection used by your application. See the following sections.
3. Within your application, set up the connection.

The steps required to prepare the client computer depend on the type of connection used by the client application. In all cases, client credentials in the form of the wallet file must be downloaded to the client.

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

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

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. Use either the full Oracle Database Client 11.2.0.4 (or higher) or the Oracle Instant Client 12.1.0.2 (or higher). The Instant Client contains the minimal software needed to make an Oracle Call Interface connection. The Instant Client 12.1.0.2 (or higher) is sufficient for most applications.
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
```

To (UNIX/Linux example):

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

To (Windows example):

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

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

**Note:**

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.

**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.

**Note:**

Connections through an HTTP proxy are only available with Oracle Client software version 12.2.0.1 or later.

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:

```
ADWC1_high =
  (description=
   (address=
    (https_proxy=proxyhostname) (https_proxy_port=80)
    (protocol=tcps) (port=1522) (host=adwc.example.oraclecloud.com)
   )
   (connect_data=(service_name=adwc1_high.adwc.oraclecloud.com)
    )
   (security=(ssl_server_cert_dn="adwc.example.oraclecloud.com,
     OU=Oracle BMCS US,O=Oracle Corporation,L=Redwood
     City,ST=California,C=US")
    )
  )
```

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 Net Services Reference.

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/adwc_credentials")))
   ```

2. Copy the entries in the tnsnames.ora file provided in the Autonomous Data Warehouse wallet to your existing tnsnames.ora file.
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 JDBC Thin Connections and Wallets for the steps required to use JDBC Thin connections to connect to an Oracle Database server.

Using Applications with Support for Wallets

Some applications allow you to choose a credentials file as part of the connection properties.

For example, in SQL Developer 18.3 and higher, in the Connection Type field select the value Cloud Wallet that allows you to enter a credentials file in the Configuration File field. SQL Developer then presents a list of the available connections in the Service field (the connections are included in the credentials files).

If your application provides support for wallets or provides specific support for an Autonomous Data Warehouse connection, for example, Oracle SQL Developer, Oracle recommends that you use that type of connection.
Download Client Credentials (Wallets)

Oracle client credentials (wallet files) are downloaded from Autonomous Data Warehouse by a service administrator. If you are not an Autonomous Data Warehouse administrator, your administrator should provide you with the client credentials.

To download client credentials you can use the Oracle Cloud Infrastructure console or the Autonomous Data Warehouse Service Console.

To download client credentials from the Oracle Cloud Infrastructure console:

1. Navigate to the Autonomous Database details page.
2. Click DB 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).
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. The password must be at least 8 characters long and must include at least 1 letter and either 1 numeric character or 1 special character. This password protects the downloaded Client Credentials wallet.
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 the Autonomous Data Warehouse Service Console:

1. From the Service Console click the Administration link.
2. Click 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. The password must be at least 8 characters long and must include at least 1 letter and either 1 numeric character or 1 special character. This password protects the downloaded Client Credentials wallet.

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** 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 the Service Console 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, use **DB Connection** on the Oracle Cloud Infrastructure console as specified above.

The zip file includes the following:

- **tnsnames.ora** and **sqlnet.ora**: Network configuration files storing connect descriptors and SQL*Net client side configuration.
- **cwallet.sso** and **ewallet.p12**: Auto-open SSO wallet and PKCS12 file. PKCS12 file is protected by the wallet password provided in the UI.
- **keystore.jks** and **truststore.jks**: Java keystore and truststore files. They are 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. The expiration date shows the date when the 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 Autonomous Data Warehouse database.
Notes:

- To invalidate database client certification keys associated with a wallet, see Rotate Wallets for Autonomous Database.
- Wallet files, along with the Database user ID and password provide access to data in your Autonomous Data Warehouse 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 with Autonomous Data Warehouse.
- The README file that contains wallet expiration information is not available in wallet zip files that were downloaded before April 2020.

Connect to Autonomous Data Warehouse Using Oracle Database Tools

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

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

Topics

- Connect with Oracle SQL Developer (18.2 or later)
- Connect with Oracle SQL Developer (earlier than Version 18.2)
- Connect with SQL*Plus
- Connect with Oracle SQLcl Cloud Connection

Connect with Oracle SQL Developer (18.2 or later)

Oracle SQL Developer is a free integrated development environment that simplifies the development and management of Oracle Database in both traditional and cloud deployments.

SQL Developer can connect to Autonomous Data Warehouse using an Oracle Wallet and contains enhancements for key Autonomous Data Warehouse features. Oracle SQL Developer provides support for wallet files using the Cloud Wallet Connection
Type (SQL Developer 18.2 shows this as **Cloud PDB**). Oracle recommends that you use version 18.2 (or later); however, earlier versions of SQL Developer will work with Autonomous Data Warehouse.

Download the latest version of Oracle SQL Developer for your platform from the Download link on this page: Oracle SQL Developer.

To create a new connection to Autonomous Data Warehouse, do the following:
Obtain your credentials to access Autonomous Data Warehouse. 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** (with 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**. For more information, see **Connect with Oracle SQL Developer (earlier than Version 18.2)**.

4. Click **Connect** to connect to the database.
Connect with Oracle SQL Developer (earlier than Version 18.2)

Oracle SQL Developer is a free integrated development environment that simplifies the development and management of Oracle Database in both traditional and cloud deployments.

SQL Developer versions after SQL Developer 17.4.1 (or later) can connect to Autonomous Data Warehouse using an Oracle Wallet and this version contains enhancements for key Autonomous Data Warehouse features. Oracle SQL Developer and later provides support for wallet files using the Cloud PDB Connection Type. Oracle recommends that you use version 18.2 (or later); however, earlier versions of SQL Developer will work with Autonomous Data Warehouse.

To create a new connection to the Autonomous Data Warehouse, do the following:

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

2. Choose the Connection Type Cloud PDB.
3. Enter the following information:

**Note:**

Versions of SQL Developer starting with 18.2.0 do not require that you enter a **Keystore Password** and do not provide this field. For more information, see Connect with Oracle SQL Developer (18.2 or later).

- **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 PDB**.
- **Configuration File**: Click **Browse**, and select the client credentials zip file.
- **Keystore Password**: Enter the password generated while downloading the client credentials from Autonomous Data Warehouse.
  
  See Download Client Credentials (Wallets).
- **Service**: Enter the database TNS name. The client credentials file includes a **tnsnames.ora** file that provides database TNS names with corresponding services.
Connect with SQL*Plus

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 Data Warehouse using SQL*Plus, 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.

2. Connect using a database user, password, and database TNS name provided in the tnsnames.ora file.

   For example:
   
   sqlplus sales_dwh@adwc1_low

   SQL*Plus: Release 18.0.0.0.0 - Production on Wed Apr 3 15:09:53 2019
   Version 18.5.0.0.0

   Copyright (c) 1982, 2018, Oracle. All rights reserved.

   Enter password:
   Last Successful login time: Wed Apr 03 2019 14:50:39 -07:00

   Connected to:
   Oracle Database 18c Enterprise Edition Release 18.0.0.0.0 - Production
   Version 18.4.0.0.0

   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 an Autonomous Data Warehouse 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 with Oracle SQLcl Cloud Connection

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

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

SQLcl can connect to the Autonomous Data Warehouse 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 18.4 Production on Wed Apr 03 15:28:40 2019

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Username? (''? sales_dwh8adwc1_low
Password? (**********?) **************
Last Successful login time: Wed Apr 03 2019 15:29:19 -07:00

Connected to:
Oracle Database 18c Enterprise Edition Release 18.0.0.0.0 - Production
Version 18.4.0.0.0

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 Autonomous Data Warehouse 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
Wallet Password: **************
```

3. Connect to the Autonomous Data Warehouse database:
SQL> connect username@servicename
password

For example:

sql /nolog

SQLcl: Release 18.4 Production on Thu Jan 24 11:29:41 2019
Copyright (c) 1982, 2019, Oracle. All rights reserved.

SQL> set cloudconfig /home/adwc/wallet_ADWC1.zip
Wallet Password: **********

SQL> connect admin@adwc1_medium

--- Note: ---

If you are connecting to an Autonomous Data Warehouse 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 Managing Concurrency and Priorities on Autonomous Data Warehouse.
For information on SQLcl, see Oracle SQLcl.

Connect with Built-in SQL Developer Web

You can access SQL Developer Web, a browser-based interface of Oracle SQL Developer, from Autonomous Data Warehouse. You can run SQL statements and scripts in a worksheet and perform other tasks on your database with SQL Developer Web.

Topics

• About SQL Developer Web
• Access SQL Developer Web as ADMIN
• Provide SQL Developer Web Access to Database Users

About SQL Developer Web

Oracle SQL Developer Web in Autonomous Data Warehouse provides a development environment and a data modeler interface for Autonomous Data Warehouse.

The main features of SQL Developer Web are:
• Run SQL statements and scripts in the worksheet
• Export data
• Design Data Modeler diagrams using existing objects

SQL Developer Web is a browser-based interface of Oracle SQL Developer and provides a subset of the features of the desktop version. See About Oracle SQL Developer Web for more information.

Access SQL Developer Web as ADMIN

SQL Developer Web is bundled with each Autonomous Data Warehouse instance.

Oracle SQL Developer Web runs in Oracle REST Data Services and access is provided through schema-based authentication. To use Oracle SQL Developer Web, you must sign in as a database user whose schema is enabled for SQL Developer Web. By default the ADMIN user is enabled to access SQL Developer Web. See Provide SQL Developer Web Access to Database Users to enable another database user's schema to access SQL Developer Web.

To access SQL Developer Web you can use the Oracle Cloud Infrastructure console or the Autonomous Data Warehouse Service Console.

To access SQL Developer Web from the Oracle Cloud Infrastructure console:

1. On the Autonomous Database details page click the Tools tab.
2. In the SQL Developer Web area, click Open SQL Developer Web.
3. In the SQL Developer Web Sign in page, enter your Username and Password.
4. Click Sign in.

This shows the SQL Developer Worksheet tab.
To access SQL Developer Web from the Autonomous Data Warehouse Service Console:

1. On the instance details page click **Service Console**.
2. Click **Development**.
3. Click **SQL Developer Web**.

4. In the SQL Developer Web Sign in page, enter your **Username** and **Password**.
5. Click **Sign in**.

This shows the SQL Developer Worksheet tab.

### Provide SQL Developer Web Access to Database Users

The **ADMIN** user can provide access to SQL Developer Web to other database users.

Database users, who are not service administrators, do not have access to the Autonomous Database service console. The **ADMIN** user provides access to SQL Developer Web by enabling access for a user and providing a URL to access SQL Developer Web.
To enable schema access to SQL Developer Web:

1. Run the following code as the ADMIN user:

   ```sql
   BEGIN
   ORDS_ADMIN.ENABLE_SCHEMA(
       p_enabled => TRUE,
       p_schema => 'schema-name',
       p_url_mapping_type => 'BASE_PATH',
       p_url_mapping_pattern => 'schema-alias',
       p_auto_rest_auth => TRUE
   );
   COMMIT;
   END;
   /
   
   where:
   - `schema-name` is the database schema name in all-uppercase.
   - `schema-alias` is an alias for the schema name to use in the URL to access SQL Developer Web.
   - `p_auto_rest_auth` specifies the REST `/metadata-catalog/` endpoint requires authorization. REST uses the metadata-catalog to get a list of published services on the schema. Set this parameter to TRUE.

2. After enabling user access for the specified schema, the ADMIN provides a user with the URL to access SQL Developer Web, as follows:
   a. Select the Autonomous Data Warehouse instance.
   b. On the instance details page click **Service Console**.
   c. Click **Development**.
   d. Right-click **SQL Developer Web** and choose **Copy URL**.
      
      The copied URL is the same as the URL the ADMIN enters to access SQL Developer Web. For example:
      
      ```url
      https://dbname.adb.us-ashburn-1.example.com/ords/admin/_sdw/?nav=worksheet
      ```

      e. To provide a user with access to SQL Developer Web you need to edit the copied URL to use the alias for the schema specified with the parameter `p_url_mapping_pattern` in step 1.

      For a user to access SQL Developer Web the part of the copied URL with "admin" is replaced with the "schema-alias".

      For example, after editing you would provide this URL for the user to login:
      
      ```url
      https://dbname.adb.us-ashburn-1.example.com/ords/schema-alias/_sdw/?nav=worksheet
      ```

3. Provide the user with the modified URL.

   To access SQL Developer Web a user pastes the URL into their browser and then enters the schema’s Username and Password in the Sign-in dialog.
JDBC Thin Connections and Wallets

Autonomous Data Warehouse mandates a secure connection that uses Transport Layer Security (TLSv1.2). Java applications that use JDBC Thin driver require either Oracle Wallet or Java KeyStore (JKS). The wallet and keystore files are included in the client credentials.zip file that is available by clicking DB Connection on the Oracle Cloud Infrastructure console.

Topics

• JDBC Thin Driver Connection Prerequisites
• Using a JDBC URL Connection String with JDBC Thin Driver
• Using a JDBC Connection with 18.3 JDBC Driver
• Connecting Using JDBC Thin Driver 12.2 or Older
• JDBC Thin Connections with an HTTP Proxy

JDBC Thin Driver Connection Prerequisites

Applications that use JDBC Thin driver require the Oracle database credentials including the Oracle wallets or Java KeyStore (JKS) files when connecting to an Autonomous Data Warehouse database.

Perform the following steps before connecting to an Autonomous Data Warehouse database:

1. **Provision Autonomous Data Warehouse:** Create an Autonomous Data Warehouse database and obtain your database credentials (username and password).

2. **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 Data Warehouse.

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 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 Connection String with JDBC Thin Driver

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 services identifiable as high, medium, and low. Each service has its own TNS alias and connection string.

See Predefined Database Service Names for Autonomous Data Warehouse for more details.

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=adw.example.oraclecloud.com))  
    (connect_data=(service_name=adw_jdbctest_high.oraclecloud.com))  
    (security=(ssl_server_cert_dn="CN=adw.oraclecloud.com,OU=Oracle US,O=Oracle Corporation,L=Redwood City,ST=California,C=US")))

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"`

---

**Using a JDBC Connection with 18.3 JDBC Driver**

Applications that use JDBC Thin driver can either use Oracle Wallets or Java KeyStore (JKS) to connect to an Autonomous Data Warehouse database.

**Using Oracle Wallet**

If you choose to use the Oracle Wallet for Java connectivity to Autonomous Data Warehouse using the 18.3 JDBC Thin Driver, do the following:

1. **Make sure that the prerequisites are met:** See [JDBC Thin Driver Connection Prerequisites](#) for more information.

2. **Verify the connection:** You can either use a Java program, a servlet, or IDEs to verify the connection to the Autonomous Data Warehouse 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 an Autonomous Data Warehouse 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.

   ```java
   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:

   ```bash
   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 Data Warehouse downloaded client credentials zip file removes the need for your application to use username/password authentication.

**Using Java KeyStore**

Follow these steps to connect to Autonomous Data Warehouse using Java KeyStore (JKS) and the 18.3 JDBC Thin Driver:

1. **Make sure that the prerequisites are met:** See JDBC Thin Driver Connection Prerequisites for more information.

2. **Ready the database details:** You can either use a Java program, a servlet, or IDEs to check the connection to Autonomous Data Warehouse 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 below. 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 Autonomous Data Warehouse database.

   ```java
   DB_URL="jdbc:oracle:thin:@dbname_high?TNS_ADMIN=/Users/test/wallet_dbname"
   ```
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’re downloading the client credentials .zip file from the Autonomous Data Warehouse service console.

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:

```java
# 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:

```java
# 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 Data Warehouse 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**

If you choose to use the Oracle Wallet for Java connectivity to Autonomous Data Warehouse using 12.2 or older JDBC Drivers, do the following:

1. **Make sure that the prerequisites are met:** See JDBC Thin Driver Connection Prerequisites for more information.
2. **Verify the connection:** You can either use a Java program, a servlet, or IDEs to verify the connection to the Autonomous Data Warehouse 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:

   ```java
   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
   java -classpath
   ./lib/ojdbc8.jar:./lib/ucp.jar:./lib/oraclepki.jar:./lib/osdt_core.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
   ```
Using Java KeyStore

Follow these steps to connect to Autonomous Data Warehouse using Java KeyStore (JKS) and the 12.2 or older JDBC Thin Driver.

1. **Make sure that the prerequisites are met**: See JDBC Thin Driver Connection Prerequisites for more information.

2. **Verify the connection**: You can either use a Java program, a servlet, or IDEs to verify the connection to the Autonomous Data Warehouse 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 below. 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 Data Warehouse 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:

   ```
   ADWC1_high =
   (description=
      (address=
        (https_proxy=proxyhostname)(https_proxy_port=80)
        (protocol=tcps)(port=1522)(host=adw.example.oraclecloud.com)
      )
    )
  
  {connect_data=(service_name=adwc1_high.adw.oraclecloud.com)

  (security=(ssl_server_cert_dn="adw.example.oraclecloud.com,OU=Oracle BMCS
  US,O=Oracle Corporation,L=Redwood City,ST=California,C=US")
  )
  }
   ```

**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.

---

**Oracle Call Interface (OCI) Connections and Wallets**

Oracle Net Services can find the location of the Autonomous Data Warehouse 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 (OCI), ODBC, and JDBC OCI Connections](#) for information on setting `WALLET_LOCATION`. 
See Download Client Credentials (Wallets) for information on downloading client credentials for Autonomous Data Warehouse.

Predefined Database Service Names for Autonomous Data Warehouse

The tnsnames.ora file provided with the credentials zip file contains three database service names identifiable as high, medium, and low. The predefined service names provide different levels of performance and concurrency for Autonomous Data Warehouse.

- **high**: The High database service provides the highest level of resources to each SQL statement resulting in the highest performance, but supports the fewest number of concurrent SQL statements. Any SQL statement in this service can use all the CPU and IO resources in your database. The number of concurrent SQL statements that can be run in this service is 3, this number is independent of the number of OCPUs in your database.

- **medium**: The Medium database service provides a lower level of resources to each SQL statement potentially resulting a lower level of performance, but supports more concurrent SQL statements. Any SQL statement in this service can use multiple CPU and IO resources in your database. The number of concurrent SQL statements that can be run in this service depends on the number of OCPUs in your database.

- **low**: The Low database service provides the least level of resources to each SQL statement, but supports the most number of concurrent SQL statements. Any SQL statement in this service can use a single CPU and multiple IO resources in your database. The number of concurrent SQL statements that can be run in this service can be up to 300 times the number of OCPUs.

Sessions in these services may get disconnected if they stay idle for more than five (5) minutes and other users' sessions require the resources consumed by the idle session. This allows resources to be freed for other active users in your database.

The following shows the details for the number of concurrent statements for each connection service.

<table>
<thead>
<tr>
<th>Database Service Name</th>
<th>Concurrent Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>high</td>
<td>3</td>
</tr>
<tr>
<td>medium</td>
<td>1.25 × OCPUs</td>
</tr>
<tr>
<td>low</td>
<td>300 × OCPUs</td>
</tr>
</tbody>
</table>

The number of OCPUs is the **CPU Core Count** shown in the Oracle Cloud console.

The following table shows sample values for a database with 16 OCPUs.

<table>
<thead>
<tr>
<th>Database Service Name</th>
<th>Number of Concurrent Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>high</td>
<td>3</td>
</tr>
<tr>
<td>medium</td>
<td>20</td>
</tr>
<tr>
<td>low</td>
<td>Up to 300 times the number of OCPUs</td>
</tr>
</tbody>
</table>
Choose whichever database service offers the best balance of performance and concurrency.

Note:
When connecting for replication purposes, use the low database service name. For example use this service with Oracle GoldenGate connections.

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 Data Warehouse, 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 Data Warehouse 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 Data Warehouse with Oracle Analytics Cloud, see Visualizing Data and Building Reports in Oracle Analytics Cloud.

Connect Applications to Autonomous Data Warehouse

Developers can use standard tools to connect to Autonomous Data Warehouse.

Topics

• Connect with Microsoft .NET and Visual Studio
• Connect with JDBC Thin Driver and UCP
• Connect with Python, Node.js, and other Scripting Languages

Connect with Microsoft .NET and Visual Studio


Oracle Data Provider for .NET (ODP.NET) provides run-time ADO.NET data access to an Autonomous Data Warehouse 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 provides database application design-time support in Visual Studio, including tools for managing Autonomous Databases in the Oracle Cloud.


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

- Oracle Data Access Components - .NET Downloads
- NuGet Gallery
- Visual Studio Marketplace

Oracle recommends using the latest provider and tools version with an Autonomous Data Warehouse database.

**Set-up Instructions**

Click the following links for instructions on how to download, install, and configure these components for use with Oracle Autonomous Database:

- Developing .NET Core Applications for Oracle Autonomous Database
- Developing .NET Framework Applications for Oracle Autonomous Database
- Deploying .NET Applications for Oracle Autonomous Database

**Note:**

These set-up instructions assume you are using TCP with SSL (TCPS) for your connections to an Autonomous Data Warehouse database, which requires using wallets. See Download Client Credentials (Wallets) for more information on using wallets.

**Connect with JDBC Thin Driver and UCP**

You can use programs with JDBC Thin driver and Universal Connection Pool (UCP) to connect to Oracle Autonomous Data Warehouse.

See JDBC Thin Connections and Wallets for more information.

**Connect with Python, Node.js, and other Scripting Languages**

You can use programs in different languages, including Python, Node.js, PHP, Ruby, R, Go, and Perl to connect to Oracle Autonomous Data Warehouse. Security is enforced using client credentials.

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.

**Install the Language Driver and Client Libraries**

To connect to Oracle Autonomous Data Warehouse from your scripting language, first install the language driver and client libraries as follows:
1. Install Instant Client or the Full Client. The minimum version supported for the Full Client is Version 11.2.0.4; for the Oracle Instant Client use version 12.1.0.2 or higher:

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

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:
**Python**: To install cx_Oracle for Python, use the instructions on the following page: [cx_Oracle Installation](#).

**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](http://php.net/downloads.php) and are also available from PECL [oci8](http://php.net/downloads.php).

**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](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](#).

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

1. Obtain client security credentials to connect to an Autonomous Data Warehouse instance. You obtain a zip file containing client security credentials and network configuration settings required to access your Autonomous Data Warehouse 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 **DB Connection**. See [Download Client Credentials (Wallets)](#).
   - **Other user (non-administrator)**: Obtain the Oracle Wallet from the administrator for your Autonomous Data Warehouse.

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

```
/or/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 (OCI), ODBC, and JDBC OCI Connections.

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, adwc_user, password, and adwc_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.

Connection and Networking Options and Features

Autonomous Database provides a number of different connection and networking options and features for connecting to an Autonomous Data Warehouse database.

**Topics**

- Using ACLs, VCNs, and Private Endpoints with Autonomous Database
- Connect with Oracle Cloud Infrastructure FastConnect
- Access Autonomous Data Warehouse with VCN Transit Routing
- Access Autonomous Data Warehouse with Service Gateway
- Enable and Disable Application Continuity
- Use Database Resident Connection Pooling with Autonomous Database
Using ACLs, VCNs, and Private Endpoints with Autonomous Database

Describes the options for restricting network access to your Autonomous Data Warehouse database by specifying access control rules or using a virtual cloud network (private access with a private endpoint).

Topics

- About Network Access Options
- Overview of Restricting Access with ACLs
- Overview of Private Endpoints

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 Autonomous Data Warehouse 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 Data Warehouse database.

  See Configure Private Endpoints with Autonomous Database for more information.
Note:

After you provision or clone your database, you cannot change the selection you make, either **Allow secure access from everywhere** or **Virtual cloud network** for the Autonomous Data Warehouse database instance. However, you can clone a new database and change the selection for the new cloned database.

Overview of Restricting Access with ACLs

When you provision or clone Autonomous Database with the **Allow secure access from anywhere** option, you can restrict network access by defining an Access Control List (ACL). You can also add, modify, or remove an ACL after you provision or clone the database.

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 Data Warehouse 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.

- 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 Configure an Access Control List with Autonomous Database for the steps for configuring an access control list, either when you provision or clone your database, or whenever you want to add, modify or remove ACLs.

Overview of Private Endpoints

When you provision or clone Autonomous Database with the Virtual cloud network option you configure network access with private endpoints. Configuring a private endpoint specifies that your Autonomous Database only allows private access.

Specifying the Virtual cloud network configuration option allows traffic only 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 Autonomous Data Warehouse database.

See Configure Private Endpoints with Autonomous Database for the steps for configuring a private endpoint, either when you provision or clone your database.

Connect with Oracle Cloud Infrastructure FastConnect

Oracle Cloud Infrastructure FastConnect provides an easy way for you to connect your on-premises network to Autonomous Data Warehouse 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 Data Warehouse with Oracle Cloud Infrastructure FastConnect see FastConnect Overview

Access Autonomous Data Warehouse with VCN Transit Routing

Oracle Cloud Infrastructure supports private access from your on-premises network to an Autonomous Data Warehouse 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 an Autonomous Data Warehouse 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 an Autonomous Data Warehouse database beyond the VCN.

After transit routing is configured, there are no special steps required to access Oracle Application Express or SQL Developer Web, or to consume RESTful services published in Oracle REST Data Services.
Access Autonomous Data Warehouse with Service Gateway

Autonomous Data Warehouse supports private access from Oracle Cloud Infrastructure resources in a VCN through a service gateway.

A service gateway allows connectivity to Autonomous Data Warehouse 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 Data Warehouse. 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 Application Express or SQL Developer Web, or to consume RESTful services published in Oracle REST Data Services.

Enable and Disable Application Continuity

Change the failover type on Autonomous Data Warehouse using the DBMS_CLOUD_ADMIN procedures to enable or to disable Application Continuity. New sessions use the new failover type from the time when you modify the current value.

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 Overview of Application Continuity for more information on Application Continuity.

Enabling Application Continuity requires Oracle Client software version 18.3 (or higher).

Application Continuity is enabled on a per database service level. You supply a value for the service_name.

1. Obtain the value for the service_name parameter to supply when you enable or disable Application Continuity.

   Find the service_name using one of the following techniques:
   
   - Look in the tnsnames.ora file in the wallet.zip file you use to connect to your service. For example:

     ```
     service_name=nvt21_adbl_high.adwc.oraclecloud.com
     ```
Use a command similar to the following to SELECT from v$services on your database and identify the service where you want to enable Application Continuity:

```
SELECT name, drain_timeout FROM v$services;
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>DRAIN_TIMEOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvt21_adb1_low.adwc.oraclecloud.com</td>
<td>0</td>
</tr>
<tr>
<td>nvt21_adb1_high.adwc.oraclecloud.com</td>
<td>0</td>
</tr>
<tr>
<td>nvt21_adb1_medium.adwc.oraclecloud.com</td>
<td>0</td>
</tr>
</tbody>
</table>

Notice the DRAIN_TIMEOUT for the high service has the value 0, indicating that Application Continuity is disabled.

2. Use DBMS_CLOUD_ADMIN.ENABLE_APP_CONT to enable Application Continuity on the service you select.

For example, to enable Application Continuity for the service named nvt21_adb1_high, run the following command:

```
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_APP_CONT(
    service_name => 'nvt21_adb1_high.adwc.oraclecloud.com'
  );
END;
/
```

Check the output of the query on v$services to verify that Application Continuity is enabled.

```
SELECT name, drain_timeout FROM v$services;
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>DRAIN_TIMEOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvt21_adb1_low.adwc.oraclecloud.com</td>
<td>0</td>
</tr>
<tr>
<td>nvt21_adb1_high.adwc.oraclecloud.com</td>
<td>300</td>
</tr>
<tr>
<td>nvt21_adb1_medium.adwc.oraclecloud.com</td>
<td>0</td>
</tr>
</tbody>
</table>

Notice the DRAIN_TIMEOUT for the high service is now a non-zero value, 300, indicating that Application Continuity is enabled.

If Application Continuity is enabled and you want to disable Application Continuity, use DBMS_CLOUD_ADMIN.DISABLE_APP_CONT. For example, if you enabled Application Continuity in the service named nvt21_adb1_high and you want to disable Application Continuity for this service, run the following command:

```
BEGIN
  DBMS_CLOUD_ADMIN.DISABLE_APP_CONT(
    service_name => 'nvt21_adb1_high.adwc.oraclecloud.com'
  );
END;
/
```

See ENABLE_APP_CONT Procedure for more information on DBMS_CLOUD_ADMIN.ENABLE_APP_CONT.
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 Autonomous Data Warehouse 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 you are using to connect to your Autonomous Data Warehouse database.
   
   See Download Client Credentials (Wallets) for more information.

2. Modify the tnsnames.ora file to add the server type SERVER=POOLED.
   
   For example:

   ```
   dbname_high= (description= 
      (address=(protocol=tcps)(port=1522) 
      (host=adw.example.oraclecloud.com)) 
      (connect_data=(service_name=example_high.oraclecloud.com) 
       (SERVER=POOLED)) 
      (security=(ssl_server_cert_dn="CN=adw.oraclecloud.com,OU=Oracle US,O=Oracle Corporation,L=Redwood City,ST=California,C=US")))
   ```

   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.
- Autonomous Database uses the following parameter values for DRCP. You cannot modify these parameter values:
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INACTIVITY_TIMEOUT</td>
<td>600</td>
<td>The maximum time, in seconds, the pooled server can stay idle in the pool. After this time, the server is terminated.</td>
</tr>
<tr>
<td>MAX_LIFETIME_SESSION</td>
<td>86400</td>
<td>The time, in seconds, to live for a pooled server in the pool.</td>
</tr>
<tr>
<td>MAX_THINK_TIME</td>
<td>300</td>
<td>The maximum time of inactivity, in seconds, for a client after it obtains a pooled server from the pool with no open transactions in it. After obtaining a pooled server from the pool, if the client application does not issue a database call for the time specified by MAX_THINK_TIME the pooled server is freed and the client connection is terminated.</td>
</tr>
</tbody>
</table>

See Using Database Resident Connection Pool for more information.

Use Database Links with Autonomous Data Warehouse

You can create database links to Oracle databases that are accessible from an Autonomous Data Warehouse database.

Topics

- Create Database Links from Autonomous Database to Other Databases
- Create Database Links from Other Databases to Autonomous Database
- Drop Database Links

Create Database Links from Autonomous Database to Other Databases

Use `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to create database links from an Autonomous Database to another database.

To use database links with Autonomous Data Warehouse 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 for an Autonomous Transaction Processing or Autonomous Data Warehouse database. Other databases must be configured to use TCP/IP with SSL (TCPS) authentication. See Configuring Secure Sockets Layer Authentication for more information.

To ensure security, the database link port is restricted to the range 1521–1525. You specify the target database port when you create a database link with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

To use database links the target database must be accessible. Some databases, including Autonomous Databases may limit access, for example using Access Control Lists. If access to the target database is limited and does not include the database where you are creating the database link, this would prevent using database links with those target databases. See Configure an Access Control List with Autonomous Database for more information.
To create database links to a target database do the following:

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. Upload the target database wallet to the `data_pump_dir` directory, or to another directory where you have write privileges, using `DBMS_CLOUD.GET_OBJECT`.

   For example:

   ```sql
   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 => 'DATA_PUMP_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 create the credentials to access the target database.

4. On Autonomous Data Warehouse 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.
For example:

BEGIN
  DBMS_CLOUDCREATE_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.

5. 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.adwc.example.oraclecloud.com',
    ssl_server_cert_dn =>
      'CN=adwc.example.oraclecloud.com,OU=Oracle BMCS FRANKFURT,O=Oracle Corporation,L=Redwood City,ST=California,C=US',
    credential_name => 'DB_LINK_CRED',
    directory_name => 'DATA_PUMP_DIR');
END;
/

Users other than ADMIN require privileges to run DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK. See CREATE_DATABASE_LINK Procedure for more information.

6. Use the database link you created to access data on the target database.

For example:

  SELECT * FROM employees@SALESLINK;

Notes for database links:

- Autonomous Database supports creating database links only if the target database is accessible through a public IP or public hostname.
• 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 `DATA_PUMP_DIR` directory. This means with a `cwallet.sso` in `DATA_PUMP_DIR` you can only create database links to the databases for which the wallet 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.

• The `DATA_PUMP_DIR` is the only available predefined directory for uploading the target database wallet with `DBMS_CLOUD.GET_OBJECT`. 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.

• Supported target database versions are: 19c, 18c, 12.2.0, 12.1.0, and 11.2.0.

**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 the `SEC_CASE_SENSITIVE_LOGON` parameter to `true` on the target database.

• To list the database links, use the `ALL_DB_LINKS` view. See ALL_DB_LINKS for more information.

• For the credentials you create in Step 4, 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.

For additional information, see:

• CREATE_DATABASE_LINK Procedure
• GET_OBJECT Procedure
• UPDATE_CREDENTIAL Procedure
Create Database Links from Other Databases to Autonomous Database

You can create database links from another Oracle Database to an Autonomous Database.

To create database links to an Autonomous Data Warehouse 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 Data Warehouse 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 Data Warehouse database. Note that the security path includes `my_wallet_directory`; the path where you unzip the Autonomous Database wallet.

```sql
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.adwc.oraclecloud.com))
      (security=(my_wallet_directory=/u01/targetwallet)
        (ssl_server_dn_match=true)
        (ssl_server_cert_dn="CN=example2.oraclecloud.com,OU=Oracle BMCS US,O=Oracle Corporation,L=Redwood City,ST=California,C=US")));
```

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:

```sql
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 Other Databases

## 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:

```sql
BEGIN
    DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK(
        db_link_name => 'SALESLINK');
END;
/
```

See `DROP_DATABASE_LINK Procedure` for detailed information about the procedure.
Loading Data with Autonomous Data Warehouse

Describes packages and tools to load data with Autonomous Data Warehouse.

Topics

• About Data Loading
• Load Data from Files in the Cloud
• Import Data Using Oracle Data Pump on Autonomous Data Warehouse
• Load Data from Local Files with SQL Developer Web
• Use Oracle GoldenGate to Replicate Data to Autonomous Data Warehouse
• Load Data from Local Files Using SQL*Loader
• Managing DML Performance and Compression

About Data Loading

You load data into Autonomous Data Warehouse using Oracle Database tools, and Oracle or other 3rd party data integration tools.

In general you load data from files local to your client computer or from files stored in a cloud-based object store. For data loading from files in the cloud, Autonomous Data Warehouse provides a new PL/SQL package, DBMS_CLOUD.

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 Autonomous Data Warehouse. 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.

For a tutorial on data loading using Oracle Cloud Infrastructure Object Storage, see Loading Your 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 with Autonomous Data Warehouse for more information.
Load Data from Files in the Cloud

The PL/SQL package DBMS_CLOUD provides support for loading data from text, Parquet, and Avro files in the Cloud to your tables in Autonomous Data Warehouse. You can also load Data Pump dump files in the Cloud to your tables in Autonomous Data Warehouse.

The package DBMS_CLOUD supports loading from files in the following cloud services: Oracle Cloud Infrastructure Object Storage, Oracle Cloud Infrastructure Object Storage Classic, Azure Blob Storage, and Amazon S3.

Topics

- Create Credentials and Copy Data into an Existing Table
- Create Credentials and Load Data Pump Dump Files into an Existing Table
- Load JSON Documents with Autonomous Database
- Monitor and Troubleshoot Loads
- List Credentials
- Delete Credentials

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 Data Warehouse 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:

```sql
SET DEFINE OFF
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adwc_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 complete details and an example using Oracle Cloud Infrastructure Object Storage credentials, see Loading Your Data Into Autonomous Data Warehouse.

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:

```sql
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.
- **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, Parquet, or Avro.

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 Parquet or Avro 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 on your Autonomous Data Warehouse 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:

```sql
SET DEFINE OFF
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME,'
    username => 'adwc_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 complete details and an example using Oracle Cloud Infrastructure Object Storage credentials, see Loading Your Data Into Autonomous Data Warehouse.

For detailed information about the parameters, see CREATE_CREDENTIAL Procedure.
2. Load data into an existing table using the procedure `DBMS_CLOUD.COPY_DATA`. For example:

```sql
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' => '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.
- `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 Parquet or Avro Files`.
Load JSON Documents with Autonomous Database

The PL/SQL package DBMS_CLOUD provides support for loading documents into collections in Autonomous Database.

Topics:
- About Loading JSON Documents
- Load a Line-Delimited JSON Document into a Collection
- Load an Array of JSON Documents into a Collection
- Monitor and Troubleshoot COPY_COLLECTION Loads

About Loading JSON Documents

You load collections into Autonomous Data Warehouse using Oracle Database Tools, and Oracle or other third-party data integration tools.

Typically, you load data from SODA collections local to your client computer or from a Cloud-based object store. For loading data from collections in the cloud, Autonomous Data Warehouse provides the procedure DBMS_CLOUD.COPY_COLLECTION.

DBMS_CLOUD.COPY_COLLECTION supports the following typical document loading procedures:
- Loading line-delimited JSON into a collection. See Load a Line-Delimited JSON Document 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.

Load a Line-Delimited JSON Document 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 Autonomous Data Warehouse database as a single document.

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:

```bash
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 Autonomous Data Warehouse 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 => 'adwc_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.

For complete details and an example using Oracle Cloud Infrastructure Object Storage credentials, see Loading Your Data Into Autonomous Data Warehouse.

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 '')
    );
```

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.
- **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. `unpackarrays`, `recorddelimiter`, `ignoreblanklines`, `characterset`, `rejectlimit` and `compression` are the formats supported while loading JSON data. Any other formats specified will result in an error. 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 collection **fruit** on your Autonomous Data Warehouse database now contains one document for each line in the file **myCollection.json**.

### Load an Array of JSON 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 documents to your database from within a JSON array.

This example uses the JSON file **fruit_array.json**. The following shows the contents of the file **fruit_array.json**:

```json
[{
"name" : "apple", "count": 20 },
{"name" : "orange", "count": 42 },
{"name" : "pear", "count": 10 }]
```

Before loading data into Autonomous Data Warehouse, 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:
  
  ```bash
  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:
  
  ```bash
  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 Autonomous Data Warehouse database:

1. Store your object store credentials using the procedure DBMS_CLOUD.CREATE_CREDENTIAL, as shown in the following example:

   ```sql
   SET DEFINE OFF
   BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DEF_CRED_NAME',
       username => 'adwc_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.

   ```sql
   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}'
   );
   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 the `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.
- `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. `unpackarrays, recorddelimiter, ignoreblanklines, characterset, rejectlimit` and `compression` are the formats supported
  while loading JSON data. Any other formats specified will result in an error.

  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.CREATE_CREDENTIAL 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`:

```sql
SELECT table_name, owner_name, type, status, start_time, update_time,
      logfile_table, badfile_table
FROM user_load_operations WHERE type = 'COPY';
```

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>OWNER_NAME</th>
<th>TYPE</th>
<th>STATUS</th>
<th>START_TIME</th>
<th>UPDATE_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRUIT</td>
<td>ADMIN</td>
<td>COPY</td>
<td>COMPLETED</td>
<td>2020-04-23 22:27:38</td>
<td>&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRUIT</td>
<td>ADMIN</td>
<td>COPY</td>
<td>FAILED</td>
<td>2020-04-23 22:28:36</td>
<td>COPY$2_LOG</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRUIT</td>
<td>ADMIN</td>
<td>COPY</td>
<td></td>
<td>2020-04-23 22:28:37</td>
<td>COPY$2_BAD</td>
</tr>
</tbody>
</table>
```
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.

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. 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>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>OWNER_NAME</th>
<th>TYPE</th>
<th>STATUS</th>
<th>START_TIME</th>
<th>UPDATE_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGFILE_TABLE</td>
<td>BADFILE_TABLE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>------</td>
<td>----------</td>
<td>--------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>CHANNELS</td>
<td>SH</td>
<td>COPY</td>
<td>COMPLETED</td>
<td>06-NOV-18 01.55.19.3</td>
<td>06-NOV-18 01.55.28.2</td>
</tr>
<tr>
<td>COPY$21_LOG</td>
<td>COPY$21_BAD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Using this SELECT statement with a WHERE clause predicate on the TYPE column, shows load operations with the type COPY.
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:

```
select * from COPY$21_LOG;
```

The column BADFILE_TABLE shows the name of the table you can query to look at the rows that got errors during loading. For example, the following query shows the rejected records for the load operation:

```
select * from COPY$21_BAD;
```

Depending on the errors shown in the log and the rows shown in the specified BADFILE_TABLE table 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 badfiles are not generated.

---

**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.

Monitor and Troubleshoot Parquet and Avro File Loading

As with other data files, 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 Parquet and Avro files, when the format parameter type is set to the value 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.
List Credentials

The PL/SQL package DBMS_CLOUD provides the ability to store your object storage credentials in the database 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;
```

<table>
<thead>
<tr>
<th>CREDENTIAL_NAME</th>
<th>USERNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADB_TOKEN</td>
<td><a href="mailto:user_name@example.com">user_name@example.com</a></td>
</tr>
<tr>
<td>double_quotes</td>
<td>&quot;Created via DBMS_CLOUD.create_credential&quot;</td>
</tr>
<tr>
<td>DEF_CRED_NAME</td>
<td><a href="mailto:user_name@example.com">user_name@example.com</a></td>
</tr>
<tr>
<td>double_quotes</td>
<td>&quot;Created via DBMS_CLOUD.create_credential&quot;</td>
</tr>
</tbody>
</table>

See ALL_CREDENTIALS for more information.

Delete Credentials

The PL/SQL package DBMS_CLOUD provides the ability to store your object storage credentials in the database 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 Summary of DBMS_CLOUD Subprograms.

Import Data Using Oracle Data Pump on Autonomous Data Warehouse

Oracle Data Pump offers very fast bulk data and metadata movement between Oracle databases and Autonomous Data Warehouse.

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 Data Warehouse.

Topics
• Export Your Existing Oracle Database to Import into Autonomous Data Warehouse
• Import Data Using Oracle Data Pump Version 18.3 or Later
• Import Data Using Oracle Data Pump (Versions 12.2.0.1 and Earlier)
• Access Log Files for Data Pump Import

Export Your Existing Oracle Database to Import into Autonomous Data Warehouse

You need to use Oracle Data Pump Export to export your existing Oracle Database schemas to migrate them to Autonomous Data Warehouse using Oracle Data Pump Import.

Oracle recommends using the following Data Pump Export parameters for faster and easier migration to Autonomous Data Warehouse:

```plaintext
eclude=index,cluster,indextype,materialized_view,materialized_view_log,materialized_zonemap,db_link
data_options=group_partition_table_data
parallel=n
schemas=schema_name
dumpfile=export\u.dmp
```

Oracle Data Pump Export provides several export modes, Oracle recommends using the schema mode for migrating to Autonomous Data Warehouse. 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 Autonomous Data Warehouse database.

The `exclude` and `data_options` parameters ensure that the object types not required in Autonomous Data Warehouse are not exported and table partitions are grouped together so that they can be imported faster during the import to Autonomous Data Warehouse. If you want to migrate your existing indexes, materialized views, and materialized view logs to Autonomous Data Warehouse and manage them manually, you can remove those object types from the exclude list which will export those object types too. Similarly, if you want to migrate your existing partitioned tables as-is without converting them into non-partitioned tables and manage them manually you can remove the `data_options` argument which will export your partitioned tables as-is. For more information, see Managing Partitions, Indexes, and Materialized Views.

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 an Autonomous Data Warehouse database with 16 CPUs:

```plaintext
gexpdp sh/sh@orcl \
exclude=index,cluster,indextype,materialized_view,materialized_view_log,materialized_zonemap,db_link \
data_options=group_partition_table_data \
parallel=16 \
schemas=sh \
```

3-14
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 Data Warehouse 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, 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.

Importing with Oracle Data Pump and Setting credential Parameter

1. Store your Cloud Object Storage credential using DBMS_CLOUD.CREATE_CREDENTIAL. For example:

   BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(  
       credential_name => 'DEF_CRED_NAME',  
       username => 'adwc_user@example.com',  
       password => 'password'  
   );
   END;
   /

   Chapter 3
   Import Data Using Oracle Data Pump on Autonomous Data Warehouse
For more information on the credentials for different Cloud Object Storage services, see **CREATE_CREDENTIAL Procedure**.

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@ADWC1_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 \\
    partition_options=merge \\
    transform=segment_attributes:n \\
    transform=dwcs_cvt_iots:y transform=constraint_use_default_index:y \\
    exclude=index,cluster,indextype,materialized_view,materialized_view_log,materialized_zonemap,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 the number of OCPUs in your Autonomous Data Warehouse as shown in the example.

For information on which database service name to connect to run Data Pump Import, see **Managing Concurrency and Priorities on Autonomous Data Warehouse**.

For the dump file URL format for different Cloud Object Storage services, see **DBMS_CLOUD Package File URI Formats**.

This example shows the recommended parameters for importing into your Autonomous Data Warehouse.

In this example:

- Partitioned tables are converted into non-partitioned tables.
- Storage attributes for tables are ignored.
- Index-organized tables are converted into regular tables.
- Indexes created for primary key and unique key constraints will be renamed to the constraint name.
- Indexes, clusters, indextypes, materialized views, materialized view logs, and zone maps are excluded during Data Pump Import.
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 Data Warehouse, see Restrictions for 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 Data Warehouse 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 Data Warehouse 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, 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 must set the `DEFAULT_CREDENTIAL` property as shown in Step 2 and supply the `default_credential` keyword as shown in Step 3. 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.

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:

```sql
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'DEF_CRED_NAME',
        username => 'adwc_user@example.com',
        password => 'password'
    );
END;
/
```

For more information on the credentials for different Cloud Object Storage services, see CREATE_CREDENTIAL Procedure.
2. Set the credential as the default credential for your Autonomous Data Warehouse, as the ADMIN user. For example:

```sql
ALTER DATABASE PROPERTY SET DEFAULT_CREDENTIAL = 'ADMIN.DEF_CRED_NAME'
```

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:

```bash
impdp admin/password@ADWC1_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 \
  partition_options=merge \
  transform=segment_attributes:n \
  exclude=index,cluster,indextype,materialized_view,materialized_view_log,materialized_zonemap,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 the number of OCPUs in your Autonomous Data Warehouse as shown in the example.

For information on which database service name to connect to run Data Pump Import, see Managing Concurrency and Priorities on Autonomous Data Warehouse.

For the dump file URL format for different Cloud Object Storage services, see DBMS_CLOUD Package File URI Formats.

This example shows the recommended parameters for importing into your Autonomous Data Warehouse.

In this example:

- Partitioned tables are converted into non-partitioned tables.
- Storage attributes for tables are ignored.
- Indexes, clusters, indextypes, materialized views, materialized view logs, and zone maps are excluded during Data Pump Import.
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 Data Warehouse, see Restrictions for SQL Commands.

For detailed information on Oracle Data Pump Import parameters see Oracle Database Utilities.

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:

```plsql
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.

For more information, see Summary of DBMS_CLOUD Subprograms.

Load Data from Local Files with SQL Developer Web

In SQL Developer Web, from the Worksheet page, you can load data from local files into an existing table.

Topics

- Load Data into Existing Autonomous Database Table with SQL Developer Web
Load Data into Existing Autonomous Database Table with SQL Developer Web

You can load data into an existing table in Autonomous Database with the SQL Developer Web import from file feature.

Before you load data, create the table in Autonomous Database. The file formats that you can upload with the SQL Developer Web upload feature are CSV, XLS, XLSX, TSV and TXT.

To upload data from local files to an existing table with SQL Developer Web, do the following:

1. Open SQL Developer Web from Autonomous Database and sign in.
   
   There are several options for how to start SQL Developer Web, depending on your schema and system privileges. See Connect with Built-in SQL Developer Web for more information.

2. To import data, in SQL Developer Web, select the Worksheet tab.
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:

![Review page](image)

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.

SQL Developer Web 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 SQL Developer Web to upload data.
Use Oracle GoldenGate to Replicate Data to Autonomous Data Warehouse

You can replicate data to Autonomous Data Warehouse using Oracle GoldenGate On Premises and Oracle GoldenGate Marketplace.

For more information, see Replicating Data to Oracle Autonomous Database.

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.

If you use SQL*Loader to load data, note that Autonomous Data Warehouse does not gather optimizer statistics for your load and you need to gather optimizer statistics manually as explained in Manage Optimizer Statistics on Autonomous Data Warehouse. Autonomous Data Warehouse 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 detailed information on SQL*Loader see, Oracle Database Utilities.

Managing DML Performance and Compression

Autonomous Data Warehouse 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:

```sql
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,
);```

Chapter 3

Use Oracle GoldenGate to Replicate Data to Autonomous Data Warehouse

3-24
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;
```
Querying External Data with Autonomous Data Warehouse

Describes packages and tools to query and validate data with Autonomous Data Warehouse.

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 with Autonomous Data Warehouse for more information.

Topics

• Query External Data
• Query External Data with Parquet or Avro Source Files
• Query External Partitioned Data
• Query Hybrid Partitioned Data
• Query External Data Pump Dump Files
• Validate External Data
• Validate External Partitioned Data
• Validate Hybrid Partitioned Data
• View Logs for Data Validation

Query External Data

To query data in files in the Cloud, you need to first store your object storage credentials in your Autonomous Data Warehouse, and then create an external table using the PL/SQL procedure DBMS_CLOUD.CREATE_EXTERNAL_TABLE.

The source file in this example, channels.txt, has the following data:
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           => 'adwc_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`.

The procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` supports external files in the supported cloud object storage services, including: Oracle Cloud Infrastructure Object Storage, Microsoft Azure, and AWS S3. 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 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 source files you want to query.
• format: defines the options you can specify to describe the format of the source file.

• 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 Package File URI Formats for more information on the supported cloud object storage services.

Query External Data with Parquet or Avro Source Files

Autonomous Data Warehouse makes it easy to access Parquet or Avro data stored in object store using external tables. 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 either Parquet or Avro types to Oracle data types. You only need to know the location of the data in object store, specify its type, Parquet or Avro, and have credentials to access the source file on your object store.
The steps to use external tables are very similar for 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 Data Warehouse, do the following:

1. Store your object store credentials, to access the object store, using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`:

   ```sql
   BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DEF_CRED_NAME',
       username => 'adwc_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 for 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, Microsoft Azure, and AWS S3. 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.

   ```sql
   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.
- 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.

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 Parquet or Avro data types. The valid schema parameter values are:

- first: Analyze the schema of the first 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 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:

```sql
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)
```

```sql
SELECT prod_id, quantity_sold, gender, city, income_level
FROM sales_extended_ext
WHERE ROWNUM < 8;
```
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 Parquet or Avro Files and COPY_DATA Procedure for Parquet or Avro Files for more information.

See DBMS_CLOUD Package File URI Formats for information on supported cloud object storage services.

Query External Partitioned Data

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.

When you create a partitioned external table, you include a partitioning clause in the DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE statement. The partitioning clause that you include depends upon the data files in the Cloud and the type of partitioning you use. See Creating a Partitioned External Table for more information.

1. Store your object store credentials using the procedure DBMS_CLOUD.CREATE_CREDENTIAL.

   For example:

   ```sql
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL (  
       credential_name => 'DEF_CRED_NAME',  
       username => 'adwc_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 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, including: Oracle Cloud Infrastructure Object Storage, Microsoft Azure, and AWS S3. The credential is a table level property; therefore, the external files must be on the same object store.

For example:

```sql
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.
- `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 Data Warehouse 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:

```sql
SELECT * FROM pet1 WHERE col1 < 750;
```
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 Package File 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.

**Note:**

Hybrid partitioned tables are only supported with Oracle Database 19c onwards.

When you 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:

   ```sql
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL (
       credential_name => 'DEF_CRED_NAME',
       username => 'adwc_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 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, including: Oracle Cloud Infrastructure Object Storage, Microsoft Azure, and AWS S3. The credential is a table level property; therefore, the external files must be on the same object store.

For example:

```sql
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.
- 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 Data Warehouse 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:

```sql
SELECT * FROM hpt1 WHERE col1 < 750;
```

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 Package File 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:

   ```sql
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DEF_CRED_NAME',
       username => 'adwc_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:

```sql
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' => 'datapump', 'rejectlimit' => '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. 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:

```sql
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 Package File URI Formats for more information on the supported cloud object storage services.

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 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',
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.

```
Note:

Hybrid partitioned tables are only supported with Oracle Database 19c onwards.
```

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:

```sql
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:

```sql
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';
```

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.
Creating Dashboards, Reports, and Notebooks with Autonomous Data Warehouse

Autonomous Data Warehouse includes built-in support for data analytics using Oracle Machine Learning (OML) a browser-based interactive data analysis environment for data scientists.

Topics

• About Creating Dashboards, Reports, and Notebooks with Oracle Machine Learning
• Work with Oracle Machine Learning for Data Access, Analysis, and Discovery
• Work with Analytics and Visualization

About Creating Dashboards, Reports, and Notebooks with Oracle Machine Learning

Oracle Machine Learning provides the following:

Collaborative Interface for real-time analytics:

• Based on Apache Zeppelin
• Allows creation of workspaces, projects, and notebooks
• Provides a browser-based interactive data analysis environment to develop, document, share, and automate analytical methodologies
• Notebooks execute in Zeppelin Servers and make secure (TCPS) JDBC connections to Autonomous Data Warehouse for data access.

OML Components:

• OML User Administration Application:
  – Web based administrative UI for managing (list, create, update, delete) Oracle Machine Learning users
  – Oracle Machine Learning users map to Autonomous Data Warehouse database users
  – Access to Oracle Machine Learning User Management is limited to the administrator

See Create and Update User Accounts for Oracle Machine Learning for more information on OML User Administration.

• OML Application
Work with Oracle Machine Learning for Data Access, Analysis, and Discovery

You can use Oracle Machine Learning to access data and for data discovery, analytics, and notebooks.

To access Oracle Machine Learning you can use the Oracle Cloud Infrastructure console or the Autonomous Data Warehouse Service Console.

To access Oracle Machine Learning from the Oracle Cloud Infrastructure console:

1. Select an Autonomous Data Warehouse instance and on the Autonomous Database details page click the **Tools** tab.

2. In the **Oracle ML User Administration** area, click **Open Oracle ML User Administration**.

3. Enter your **Username** and **Password**.

   The Administrator creates OML users. See **Create and Update User Accounts for Oracle Machine Learning** for details.

4. Click **Sign in**.
To access Oracle Machine Learning from the Autonomous Data Warehouse Service Console:

1. Select an Autonomous Data Warehouse instance and on the Autonomous Database details page click **Service Console**.

2. Click **Development**.

3. On the Development page click **OML Notebooks**.

4. Enter your username and password.
   
   The Administrator creates OML users. See [Create and Update User Accounts for Oracle Machine Learning](#) for details.

5. Click **Sign In**.

   This shows Oracle Machine Learning user application.
Oracle Machine Learning allows you to access your data in Autonomous Data Warehouse 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 Autonomous Data Warehouse.

**Work with Analytics and Visualization**

Gain insight into your data with Oracle Analytics Cloud and Oracle Analytics Desktop. These tools let you explore your Autonomous Data Warehouse data through advanced analytics and interactive visualizations.

**Topics**

- Using Oracle Analytics Cloud with Autonomous Data Warehouse
- Working with Oracle Analytics Desktop in Autonomous Data Warehouse

**Using Oracle Analytics Cloud with Autonomous Data Warehouse**

You can use Oracle Analytics Cloud with Autonomous Data Warehouse.

Built on a high-performance platform with flexible data storage, 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 premises 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.

For information on connecting from Oracle Analytics Cloud to Autonomous Data Warehouse, see [Connect with Oracle Analytics Cloud](#).

You can find out more information on Oracle Analytics Cloud in *Visualizing Data and Building Reports in Oracle Analytics Cloud*.
Working with Oracle Analytics Desktop in Autonomous Data Warehouse

Gain insight into your data with Oracle Analytics Desktop. Oracle Analytics Desktop lets you explore your Autonomous Data Warehouse 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 Data Warehouse data so you can focus on exploring interesting data patterns. Just connect to Autonomous Data Warehouse, 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.

For information on connecting Oracle Analytics Desktop to Autonomous Data Warehouse, see Connect with Oracle Analytics Desktop.

To download Oracle Analytics Desktop, see Oracle Data Visualization Desktop.
Moving Data from Autonomous Data Warehouse to Other Oracle Databases

Oracle Data Pump offers very fast bulk data and metadata movement between Autonomous Database and other Oracle databases.

To export data from your Autonomous Data Warehouse database to other Oracle databases, use one of the following methods:

• Use Oracle Data Pump to export the data from your Autonomous Data Warehouse database to a directory on your database, and then move the data from the directory to your cloud object store.
• Use Oracle Data Pump to export the data to your cloud object store 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` This method is only supported with Oracle Cloud Infrastructure Object Storage and Oracle Cloud Infrastructure Object Storage Classic.

Topics

• Move Data with Data Pump Export to an Autonomous Database Directory
• Move Data with Data Pump Export to Object Store
• Move Selective Data to Object Store Using `DBMS_CLOUD.EXPORT_DATA`
• Download Dump Files, Run Data Pump Import, and Clean Up Object Store
• Access Log Files for Data Pump Export

Move Data with Data Pump Export to an Autonomous Database Directory

To move data from your Autonomous Data Warehouse database to other Oracle databases you can use Oracle Data Pump to first 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
This export method using Oracle Data Pump includes the following steps:

1. Use Data Pump Export to save a dump file set to a directory on your Autonomous Data Warehouse database.
   See Use Data Pump to Create a Dump File Set on Autonomous Data Warehouse for details.

2. Move the dump file set from the directory on your Autonomous Data Warehouse database to your cloud object store.
   See Move Dump File Set from Autonomous Data Warehouse 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 Data Warehouse

Shows the steps to export data from your Autonomous Data Warehouse database to a directory with Oracle Data Pump.

Oracle recommends using the latest Oracle Data Pump version for exporting data from Autonomous Data Warehouse 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:

The Autonomous Data Warehouse Service Console provides a link for Oracle Instant Client. To access this link from the Service Console click Development and select 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 Data Warehouse named ADWC1 with 16 OCPUs:

expdp sales/password@ADWC1_high
directory=data_pump_dir
dumpfile=exp%U.dmp
parallel=16
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 the number of OCPUs in your Autonomous Data Warehouse. For information on which database service name to connect to run Data Pump Export, see Managing Concurrency and Priorities on Autonomous Data Warehouse.

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:

<table>
<thead>
<tr>
<th>OBJECT_NAME</th>
<th>BYTES</th>
<th>CHECKSUM</th>
<th>CREATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>exp01.dmp</td>
<td>12288</td>
<td>12-NOV-19 06.10.47.0 PM GMT</td>
<td>12-NOV-19...</td>
</tr>
<tr>
<td>exp02.dmp</td>
<td>8192</td>
<td>12-NOV-19 06.10.48.0 PM GMT</td>
<td>12-NOV-19...</td>
</tr>
<tr>
<td>exp03.dmp</td>
<td>1171456</td>
<td>12-NOV-19 06.10.48.0 PM GMT</td>
<td>12-NOV-19...</td>
</tr>
<tr>
<td>exp04.dmp</td>
<td>348160</td>
<td>12-NOV-19 06.10.48.0 PM GMT</td>
<td>12-NOV-19...</td>
</tr>
<tr>
<td>export.log</td>
<td>1663</td>
<td>12-NOV-19 06.10.50.0 PM GMT</td>
<td>12-NOV-19...</td>
</tr>
</tbody>
</table>

2. Move the dump file set to your cloud object store. See Move Dump File Set from Autonomous Data Warehouse to Your Cloud Object Store for details.
Move Dump File Set from Autonomous Data Warehouse to Your Cloud Object Store

To move the dump file set to your Cloud Object Store, upload the files from the Autonomous Data Warehouse database directory to your Cloud Object Store.

1. Connect to your Autonomous Data Warehouse database.

2. Store your object store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`.
   
   For example:

   ```sql
   BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
   credential_name => 'DEF_CRED_NAME',
   username => 'adwc_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 Autonomous Data Warehouse 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 your Autonomous Data Warehouse database to other Oracle databases you can use Oracle Data Pump to directly export to your object store. Using this export method you use Oracle Data Pump to directly export 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 an Autonomous Data Warehouse database to other Oracle databases, do the following:

1. **Use Data Pump Export to export from an Autonomous Data Warehouse database to Object Storage.**

   There are differences depending on the version of Oracle Data Pump you use, Version 19.7 (or later) or Version 19.6 (and earlier):
Use Oracle Data Pump to Export Data to Object Store (Version 19.6 and Earlier)

Shows the steps to export data from your Autonomous Data Warehouse database to Object Storage with Oracle Data Pump.

Oracle recommends using the latest Oracle Data Pump version for exporting data from Autonomous Data Warehouse 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:
The Autonomous Data Warehouse Service Console provides a link for Oracle Instant Client. To access this link from the Service Console click Development and select Download Oracle Instant Client.

Exporting with Oracle Data Pump Versions 19.6 and Earlier

1. Connect to your Autonomous Data Warehouse 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 Autonomous Data Warehouse database. For example:

   ALTER DATABASE PROPERTY SET DEFAULT_CREDENTIAL = 'ADMIN.DEF_CRED_NAME'

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@ADWC1_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 the number of OCPUs in your Autonomous Data Warehouse instance, as shown in the example.

For information on which database service name to connect to run Data Pump Export, see Managing Concurrency and Priorities on Autonomous Data Warehouse.

For the dump file URL format for different Cloud Object Storage services, see DBMS_CLOUD Package File URI Formats.

This example shows the recommended parameters for exporting from Autonomous Data Warehouse. 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:**

The Oracle Cloud Infrastructure Object Store console shows multiple files for each dump file that you export, and the size of the actual dump files will be displayed as zero (0). For example:

```
exp01.dmp
exp01.dmp_aaaaaa
exp02.dmp
exp02.dmp_aaaaa
```

Oracle Data Pump divides each dump file into smaller chunks for faster uploads. Downloading the zero byte dump file from the console or Oracle Cloud Infrastructure CLI does not give you the full dump file. To download the dump files from the Object Store, use a tool that supports Swift such as `curl`, and provide your user login and Swift auth token. For example, `curl` with `GET`:

```
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.

Use Oracle Data Pump to Export Data to Object Store (Version 19.7 or Later)

Shows the steps to export data from your Autonomous Data Warehouse database to Object Storage with Oracle Data Pump.

Oracle recommends using the latest Oracle Data Pump version for exporting data from Autonomous Data Warehouse 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:

The Autonomous Data Warehouse Service Console provides a link for Oracle Instant Client. To access this link from the Service Console click Development and select Download Oracle Instant Client.

If you are using Oracle Data Pump Version 19.7 or later, then you can use the credential parameter as shown in these steps. For instructions for using Oracle Data Pump Versions 19.6 and earlier, see Use Oracle Data Pump to Export Data to Object Store (Version 19.6 and Earlier).

Exporting with Oracle Data Pump and Setting credential Parameter

1. Connect to your Autonomous Data Warehouse database.

2. Store your Cloud Object Storage credential using DBMS_CLOUD.CREATE_CREDENTIAL. For example:

   ```sql
   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@ADWC1_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.

For the best export performance use the HIGH database service for your export connection and set the `PARALLEL` parameter to the number of OCPUs in your Autonomous Data Warehouse instance, as shown in the example.

For information on which database service name to connect to run Data Pump Export, see [Managing Concurrency and Priorities on Autonomous Data Warehouse](#).

For the dump file URL format for different Cloud Object Storage services, see [DBMS_CLOUD Package File URI Formats](#).

This example shows the recommended parameters for exporting from Autonomous Data Warehouse. 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:

The Oracle Cloud Infrastructure Object Store console shows multiple files for each dump file that you export, and the size of the actual dump files will be displayed as zero (0). For example:

exp01.dmp
exp01.dmp_aaaaaa
exp02.dmp
exp02.dmp_aaaaaa

Oracle Data Pump divides each dump file into smaller chunks for faster uploads. Downloading the zero byte dump file from the console or Oracle Cloud Infrastructure CLI does not give you the full dump file. To download the dump files from the Object Store, use a tool that supports Swift such as curl, and provide your user login and Swift auth token. For example, curl with GET:

```bash
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.
Move Selective Data to Object Store Using **DBMS_CLOUD.EXPORT_DATA**

You can export data to Oracle Data Pump dump files by specifying a query.

**Note:**

To export a schema to another database, use one of the alternative methods. See Moving Data from Autonomous Data Warehouse to Other Oracle Databases for more information.

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 Autonomous Data Warehouse database.**
2. **Store your object store credentials using the procedure** **DBMS_CLOUD.CREATE_CREDENTIAL**.

   For example:

   ```sql
   BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => ©DEF_CRED_NAME©,
       username => ©adwc_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. **Export data from Autonomous Database to your Cloud Object Store as Oracle Data Pump dump file(s) by calling** **DBMS_CLOUD.EXPORT_DATA**. For example:

   ```sql
   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/export1.txt',
       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 instance on Shared Infrastructure, 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 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 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 or to an Autonomous Transaction Processing
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. The Oracle Cloud Infrastructure Object Store console shows multiple
   files for each dump file that you export, and the size of the actual dump files will be
displayed as zero (0). For example:

   exp01.dmp
   exp01.dmp_aaaaaa
   exp02.dmp
   exp02.dmp_aaaaaa

   Oracle Data Pump divides each dump file into smaller chunks for faster uploads.
   Downloading the zero byte dump file from the console or Oracle Cloud
   Infrastructure CLI does not give you the full dump file. To download the dump files
   from the Object Store, use a tool that supports Swift such as curl, and provide
   your user login and Swift auth token. For example, curl with GET:

   curl -O -v -X GET -u 'user1@example.com:auth_token' \
   https://swiftobjectstorage.us-ashburn-1.oraclecloud.com/v1/namespace-
   string/bucketname/exp01.dmp

2. Run Data Pump Import to import the dump file set to the target database.

   If you are importing the data to another Autonomous Data Warehouse database,
   see Import Data Using Oracle Data Pump on Autonomous Data Warehouse.

   If you are importing the data to an Autonomous Transaction Processing database,
   see Import Data Using Oracle Data Pump on Autonomous Transaction Processing.
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:

```sql
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.
Developing RESTful Services in Autonomous Database

You can develop and deploy RESTful Services with native Oracle REST Data Services (ORDS) support on an Autonomous Data Warehouse database. Autonomous Data Warehouse also supports SODA for REST; this allows you to use Autonomous Data Warehouse database as a simple JSON document store.

Topics

• About Oracle REST Data Services in Autonomous Database
• Access RESTful Services and SODA for REST
• Develop with Oracle REST Data Services on Autonomous Database
• Use SODA for REST with Autonomous Database
• 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 modern REST interfaces for relational data in the Autonomous Data Warehouse database. A mid-tier Java application, ORDS maps HTTP(S) verbs (GET, POST, PUT, DELETE, and so on) to database transactions and returns any results formatted using JSON.

Note:

The Oracle REST Data Services (ORDS) application in Autonomous Data Warehouse 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 OCPUs). It is not possible to change ORDS configuration or connect a customer-managed ORDS application to Autonomous Data Warehouse.

See Oracle REST Data Services for information on using Oracle REST Data Services.

See Predefined Database Service Names for Autonomous Data Warehouse for information on the low database service.
Access RESTful Services and SODA for REST

Each Autonomous Data Warehouse instance includes Oracle REST Data Services (ORDS) that provides HTTPS interfaces for working with the contents of your Oracle Database in REST enabled schemas.

To use Oracle REST Data Services and SODA for REST:

- Sign in to your Oracle Cloud Account at cloud.oracle.com.
- From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
- On the Autonomous Databases page select an Autonomous Data Warehouse instance from the links under the Display Name column.

1. On the Autonomous Database details page click Service Console.
2. Click Development.
3. The RESTful Services and SODA card shows the base URL.
4. Click **Copy URL** to copy the URL.

## Develop with Oracle REST Data Services on Autonomous Database

Autonomous Data Warehouse supports Oracle REST Data Services (ORDS).

Developing RESTful services is easy with either of the following development interfaces:

- **SQL Developer (desktop):** With SQL Developer on your desktop, you can connect to your Autonomous Data Warehouse 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 with Oracle SQL Developer (18.2 or later)](#) for more information.

- **Oracle Application Express (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 Data Warehouse **ADMIN** account is REST Enabled. This allows for REST Services to be published in the **ADMIN** schemas and allows you to access SQL Developer Web 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 Data Warehouse supports Simple Oracle Document Access (SODA) for REST.

Simple Oracle Document Access (SODA) for REST is a predeployed REST service that can be used to store JSON documents in an Autonomous Data Warehouse 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:

/ords/schema/soda/latest/*

Where schema corresponds to the REST enabled database schema (for example, "admin").

The following examples use the cURL command line tool (http://curl.haxx.se/) to submit REST requests to the Autonomous Data Warehouse database. However, other 3rd party REST clients and libraries should work as well.

This command creates a new collection named “fruit” in the ADMIN schema:

```bash
> curl -X PUT -u 'ADMIN:password' "https://rzsf8o3up2w8rzc-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/fruit"
```

These commands insert three JSON documents into the fruit collection:

```bash
> curl -X POST -u 'ADMIN:password' -H "Content-Type: application/json" --data '{"name":"orange", "count":42}' "https://rzsf8o3up2w8rzc-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://rzsf8o3up2w8rzc-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,"...}
```
This example retrieves a stored JSON document from the collection:

```bash
> curl -X POST -u 'ADMIN:password' \
-H "Content-Type: application/json" --data '{"name":"orange"}' \
"https://rzsf8o3up2w8rzc-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/fruit?action=query"
```

And finally, the following sample SQL query accesses the fruit collection:

```sql
SELECT f.json_document.name, f.json_document.count, f.json_document.color FROM fruit f;
```

This query returns three rows:

<table>
<thead>
<tr>
<th>name</th>
<th>count</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>orange</td>
<td>42</td>
<td>null</td>
</tr>
<tr>
<td>pear</td>
<td>5</td>
<td>null</td>
</tr>
<tr>
<td>apple</td>
<td>12</td>
<td>red</td>
</tr>
</tbody>
</table>

These examples show a small subset of the SODA and SQL/JSON features. For more information see:

See SODA for REST for information on Simple Oracle Document Access (SODA).

See SODA for REST HTTP Operations for information on the SODA for REST HTTP operations.
About Customer Managed Oracle REST Data Services on Autonomous Database

When you provision an Autonomous Database instance, by default Oracle REST Data Services (ORDS) is preconfigured and available for the instance. With the default ORDS, Oracle performs any required configuration, patching, and maintenance. Additionally, you can also configure Autonomous Database to use ORDS running in a customer managed environment.

When you use the default ORDS on Autonomous Database, you cannot modify any of the ORDS configuration options. For example, with the default configuration, the JDBC connection pools have a maximum of 100 connections and the connections for ORDS are preconfigured to use the LOW database service. 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 in the customer managed environment. 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.

Note:

Oracle REST Data Services 19.4.6 or higher is required to use a customer managed environment for ORDS with Autonomous Database.
You can create applications with Oracle Application Express on Autonomous Data Warehouse.

Topics

• About Oracle Application Express
• Access Oracle Application Express Administration Services
• Create Oracle Application Express Workspaces in Autonomous Data Warehouse
• Access Oracle Application Express App Builder
• Create Oracle Application Express Developer Accounts
• Use JSON Data with Oracle Application Express
• Use Web Services with Oracle Application Express
• Send Email from Oracle Application Express
• Restrictions and Limitations for Oracle Application Express with Autonomous Data Warehouse

About Oracle Application Express

Oracle Application Express (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 Application Express 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 Application Express specific to working on Autonomous Data Warehouse.

Access Oracle Application Express Administration Services

Each Autonomous Data Warehouse instance includes a dedicated instance of Oracle Application Express; 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 Application Express Administration Services.

To access Oracle Application Express Administration Services:

- Sign in to your Oracle Cloud Account at cloud.oracle.com.
- From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
- On the Autonomous Databases page select an Autonomous Data Warehouse instance from the links under the **Display Name** column.

1. To access Application Express Administration Services you can use the Oracle Cloud Infrastructure console or the Autonomous Data Warehouse Service Console.

   To access Application Express Administration Services from Autonomous Data Warehouse Service Console:
   
   a. On the Autonomous Database details page click **Service Console**.
   
   b. Click **Development**.
   
   c. Click **APEX**.

   To access Application Express Administration Services from the Oracle Cloud Infrastructure console:
   
   a. On the Autonomous Database details page click the **Tools** tab.
   
   b. In the Oracle Application Express area, click **Open APEX**.

   The Application Express Administration Services sign-in page appears.
Note:

If you already created a workspace, the Application Express workspace sign-in page appears instead. To open Administration Services, click Administration Services link.

2. In the **Password** field, enter the password for the Autonomous Data Warehouse ADMIN user.

See [Change the Administrator Password in Autonomous Data Warehouse](#) to change the password.

3. 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 Application Express Workspaces in Autonomous Data Warehouse](#) for more information.

   You can also use Administration Services to manage your Application Express instance. See Oracle Application Express Administration Services in [Oracle Application Express Administration Guide](#) for more information.
Create Oracle Application Express Workspaces in Autonomous Data Warehouse

An Autonomous Data Warehouse instance does not have any precreated workspaces for Oracle Application Express. Create a workspace if you have not already done so or use these instructions to create additional workspaces.

To create an Oracle Application Express workspace:

1. Sign in to Application Express Administration Services.
   See Access Oracle Application Express 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 Create Users with Autonomous Data Warehouse 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 Application Express App Builder and Create Oracle Application Express Developer Accounts to create additional developer accounts.
Access Oracle Application Express App Builder

Use App Builder to create and manage Oracle Application Express applications and application pages. The App Builder home page displays all installed applications in the current Oracle Application Express workspace.

To access Oracle Application Express App Builder:

1. Sign in to Application Express using the workspace name, username, and password you specify when you create the workspace.
2. On the Workspace home page, click the App Builder icon.

See Create Oracle Application Express Developer Accounts to create developer accounts.

Create Oracle Application Express Developer Accounts

Oracle Application Express 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. These steps show you how to create additional developer accounts for members of your team. When you create a developer account, a corresponding database user is automatically created.

To create developer accounts and provide direct access to Application Express:

1. Sign in to Application Express using the workspace name, username, and password you specified when you created the workspace.
2. Pull down the Administration menu in the upper right of any page and choose Manage Users and Groups.
3. Click Create User.
4. In the Username field, enter a username.
5. In the Email Address field, enter an email address.
6. (Optional) Use the on-screen and in-line help to fill in additional fields.
7. In the User is a developer field, select Yes.
8. In the Password field, enter a strong password. See Create Users with Autonomous Data Warehouse to learn more about the default password complexity rules.
9. In the Confirm Password field, confirm the password.
10. Click Create User.

To share sign-in details with developers:

1. Select an Autonomous Data Warehouse instance.
2. On the instance details page click Service Console.
3. Click Development.
4. Right-click APEX and choose Copy URL.
5. 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 Application Express environment without having to navigate to the Autonomous Data Warehouse Service Console.

**Note:**

Changing the password of Workspace Administrators and Developers through Manage Users and Groups 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 SQL Developer Web or another client to change the password of the corresponding database user.

See Access Oracle Application Express App Builder to access Oracle Application Express App Builder.

See Workspace and Application Administration in Oracle Application Express Administration Guide for more information.

### Use JSON Data with Oracle Application Express

To use Oracle Application Express for creating applications that have JSON data, you must first create views of JSON data using JSON Data Guide or the json_table SQL/JSON function.

**Topics:**

- Create A View with JSON Data Guide
- Create A View with the json_table Function

See Managing SODA Collections for more information on managing SODA collections through Oracle Application Express.

#### Create A View with JSON Data Guide

Oracle Application Express 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 Application Express SQL Workshop to create a view named `myview`:

```sql
-- 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 the json_table Function

You can also create a view on JSON data using the json_table 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 on JSON Data using SQL.

Use Web Services with Oracle Application Express

You can interact with both SOAP and RESTful style web services from Application Express in your Autonomous Data Warehouse 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 Web Source Modules, Application Express 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 Application Express 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 Application Express 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 Application Express with Autonomous Data Warehouse:

- All web services must be secured. Only HTTPS services are supported on the default port (443). Your Application Express instance is pre-configured 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. This Oracle Wallet is centrally managed and therefore you cannot consume 3rd party web services that are protected using self-signed SSL certificates.

- All web services must be accessible over the internet. The Autonomous Data Warehouse database is unable to reach web services deployed on private subnets or behind on-premises firewalls.

- Each Autonomous Data Warehouse instance is preconfigured with a network access control list (ACL) to permit outbound web service calls from Application Express. No further configuration by application developers is necessary.

- Your Application Express instance does not require an outbound web proxy.

- There is a default limit of 50,000 outbound web service requests per Application Express 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 the default limit up to 250,000 outbound web service requests by setting a value for the MAX_WEBSERVICE_REQUESTS parameter. For example, to change the limit to 250,000, connect to your Autonomous Data Warehouse 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 Application Express API Reference
Send Email from Oracle Application Express

You can use the APEX_MAIL package to send emails from Oracle Application Express applications deployed in Autonomous Data Warehouse.

Before you use APEX_MAIL you must configure an email provider in your Application Express instance. Currently, the only supported email provider is Oracle Cloud Infrastructure Email Delivery service.

Note:
Currently, third-party email providers are not supported.

To enable APEX_MAIL functionality in your Application Express instance in Autonomous Data Warehouse:

1. Identify the SMTP connection endpoint for Email Delivery. You configure the endpoint as the SMTP Host in your Application Express 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 Application Express 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 Data Warehouse 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.

```sql
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:

```sql
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 Application Express 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 limit of 5,000 emails per workspace in a 24-hour period. Oracle Cloud Infrastructure Email Delivery may impose additional limitations.

For more information, see:

- **Overview of the Email Delivery Service**
- **APEX_MAIL in Oracle Application Express API Reference**
- **APEX_INSTANCE_ADMIN in Oracle Application Express API Reference**

**Restrictions and Limitations for Oracle Application Express with Autonomous Data Warehouse**

This section lists the feature restrictions and limitations of Oracle Application Express when used within the context of Autonomous Data Warehouse. Certain limitations are required to protect the security and performance of your Oracle Application Express environment.
• Application Express Administration Services: Certain Application Express instance configuration options are disabled. The following are examples of configuration options that have been predefined by Oracle and cannot be altered:
  – Authentication scheme used to access App Builder ("Database Accounts")
  – Ability to submit and approve self-service workspace requests and change requests
  – Ability to create additional instance administrator users. Only the ADMIN user is permitted to access Administration Services. Other instance administrator users cannot be added.
  – Ability to create workspace administrator and developer users. New users can be created only in Workspace Administration
  – Web proxy, Oracle Wallet, and print server configuration
  – Daily limits of outbound web service calls and email messages
  – An option to make insecure outbound web service calls

• The following application authentication schemes are not supported:
  – HTTP Header Variable
  – LDAP Directory
  – Oracle Application Server Single Sign-On

• PDF, Excel, and Word printing options are disabled. You may be able to configure a 3rd party print server within Application Express apps.

• Only the following procedures and functions of the APEX_INSTANCE_ADMIN package are supported. See APEX_INSTANCE_ADMIN in Oracle Application Express API Reference for more information:

  ADD_SCHEMA
  ADD_WORKSPACE
  GET_PARAMETER
  REMOVE_SCHEMA
  REMOVE_WORKSPACE
  SET_PARAMETER

• Only SMTP Application Express instance parameters may be set using the APEX_INSTANCE_ADMIN package. See Available Parameter Values in Oracle Application Express API Reference for more information. Other instance parameters cannot be changed.

• Oracle Application Express is only available as a Full Development environment. Converting into a Runtime environment is not supported.

• Vanity URLs or custom domain names are not supported.

• Oracle Cloud Infrastructure Web Application Firewall (WAF) service is not supported.
Creating and Managing Directories on Autonomous Database

Each Autonomous Data Warehouse database includes a predefined `data_pump_dir` directory where you can place files. For example, you can use this 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. Use `DBMS_CLOUD.LIST_FILES` to list the contents of a directory.

Topics:
- Create Directory in Autonomous Database
- Drop Directory in Autonomous Database
- List Contents of Directory in Autonomous Database
- Copy Files Between Object Store and a Directory in Autonomous Database

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.
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:

  ```sql
  CREATE OR REPLACE DIRECTORY ROOT_DIR AS '';
  ```

  Then list the contents of `ROOT_DIR` with the following command:

  ```sql
  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. The Autonomous Data Warehouse 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 `stage` directory, run the following query:

```sql
SELECT * FROM DBMS_CLOUD.LIST_FILES('STAGE');
```

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.

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:

```sql
BEGIN
  DBMS_CLOUD.GET_OBJECT(
    credential_name => 'DEF_CRED_NAME',
    ```
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 PUT_OBJECT Procedure for more information.
10

Sending Mail with Email Delivery on Autonomous Database

Describes how to send mail using UTL_SMTP on Autonomous Database.

Topics:
- Send Mail Using Email Delivery on Autonomous Database
- SMTP Send Mail Sample Code

Send Mail Using Email Delivery on Autonomous Database

Describes the steps to send email using UTL_SMTP on Autonomous Database.

Note:

The only supported email provider is 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
   END
5. Create a PL/SQL procedure to send email.
   For example, see the sample code shown in SMTP Send Mail Sample Code.

6. Send a test email using the PL/SQL procedure you created in step 5.
   For example:

   ```plsql
   execute send_mail('taylor@example.com', 'Email from Oracle Autonomous Database', 'Sent using UTL_SMTP');
   ```

See UTL_SMTP for information on UTL_SMTP.

See Restrictions and Notes for Database PL/SQL Packages for UTL_SMTP restrictions with Autonomous Database.

SMTP Send Mail Sample Code

Shows sample code for sending mail with UTL_SMTP on Autonomous Database.

```sql
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, msg_text);```

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:

- **username**: specifies the SMTP credential username from Step 2 in Send Mail Using Email Delivery on Autonomous Database.
- **passwd**: specifies the SMTP credential password from Step 2 in Send Mail Using Email Delivery on Autonomous Database.
- **msg_from**: specifies one of the approved senders from Step 3 in Send Mail Using Email Delivery on Autonomous Database.
- **mailhost**: specifies the SMTP Connection Endpoint from Step 1 in Send Mail Using Email Delivery on Autonomous Database.
Use Oracle Extensions for IDEs to Develop Applications

Oracle extensions enable developers to connect to, browse, and manage Autonomous Databases in Oracle Cloud directly from their IDEs.

Topics

• Use Oracle Cloud Infrastructure Toolkit for Eclipse
• Use Oracle Developer Tools for Visual Studio
• Use Oracle Developer Tools for VS Code

Use Oracle Cloud Infrastructure Toolkit for Eclipse

Oracle Cloud Infrastructure Toolkit for Eclipse is a plug-in that enables Java developers to easily connect to Oracle Autonomous Database through their IDE. The plug-in is free and is available for Linux, UNIX, Microsoft Windows, and Apple Mac OS.

You can use the plug-in to perform database management operations right from Eclipse, such as creating Autonomous Databases, stopping and starting, scaling up and down, and so on. You can also use the plug-in 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 plug-in 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 Shared 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 databases
- Scale up/down Autonomous Database resources
- Restore from backup
- Update instance credentials, update the license type used
- Rotate wallets
- Convert Always Free Autonomous Database databases into paid databases

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 JSON Documents with Autonomous Database

Autonomous Database has full support for data represented as JSON documents. In an Autonomous Data Warehouse database, JSON documents can coexist with relational data.

You can access and manage JSON documents in an Autonomous Data Warehouse database 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.

Topics:

• Work with Simple Oracle Document Access (SODA) in Autonomous Database
• Work with JSON Documents Using SQL and PL/SQL APIs on Autonomous Database
• Load JSON Documents with Autonomous Database

Work with Simple Oracle Document Access (SODA) in Autonomous Database

Simple Oracle Document Access (SODA) is a set of NoSQL-style APIs that allow you to 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.

To get started with SODA, see the following:

• Overview of SODA
• Overview of SODA Filter Specifications (QBEs)

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 can be accessed from almost any programming language as SODA for REST maps SODA operations to Uniform Resource Locator (URL) patterns.

Depending on the API you want to work with, see the following:
<table>
<thead>
<tr>
<th>SODA API</th>
<th>Download and Installation</th>
<th>More Information</th>
</tr>
</thead>
</table>
| 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. | Using 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**  
SODA for C Prerequisites  
**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. | Using SODA for C |
| SODA for PL/SQL | No need to download. This is included with Autonomous Database.  
SODA for PL/SQL Prerequisites | Using 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 |
## 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 Autonomous Data Warehouse database. You can combine your JSON data with non-JSON data; then, using Autonomous Database features such as Oracle Machine Learning, 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

The PL/SQL package DBMS_CLOUD provides support for loading documents into collections in Autonomous Database.

**Topics:**

- About Loading JSON Documents
- Load a Line-Delimited JSON Document into a Collection
- Load an Array of JSON Documents into a Collection
Managing and Monitoring Autonomous Data Warehouse

This part provides information on managing Autonomous Data Warehouse.

Topics

• Starting, Stopping, and Scaling Autonomous Data Warehouse
• Managing Users on Autonomous Data Warehouse
• Managing and Monitoring Performance of Autonomous Data Warehouse
• Backing Up and Restoring Autonomous Data Warehouse
• Cloning, Moving, or Upgrading an Autonomous Data Warehouse Database
• Wallet Security, Data Safe, and Oracle Database Vault
• Configuring Network Access with Access Control Rules (ACLs) and Private Endpoints
• Maintenance Windows, Work Requests and Managing Events and Notifications
• Using a Standby Database with Autonomous Database
Starting, Stopping, and Scaling Autonomous Data Warehouse

This section describes starting, stopping, restarting, and terminating an Autonomous Database instance and also covers scaling processing power and scaling storage capacity.

Topics

- Provision Autonomous Data Warehouse
- Start Autonomous Data Warehouse
- Stop Autonomous Data Warehouse
- Restart Autonomous Data Warehouse
- Terminate Autonomous Data Warehouse
- Add CPU or Storage Resources or Enable Auto Scaling
- Remove CPU or Storage Resources or Disable Auto Scaling
- Use Auto Scaling

Provision Autonomous Data Warehouse

Follow these steps to provision a new Autonomous Data Warehouse instance using the Oracle Cloud Infrastructure Console.

1. Sign in to your Oracle Cloud Account at cloud.oracle.com.
2. Open the Oracle Cloud Infrastructure console by clicking the next to Oracle Cloud.
3. From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
4. Choose your region. See Switching Regions for information on switching regions and working in multiple regions.
5. Click Create Autonomous Database.
6. Provide basic information for the Autonomous Database.
   - Choose a 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.
   - Database name Specify the database name; it must consist of letters and numbers only. The maximum length is 14 characters.
4. Choose a workload type. Select the workload type for your database from the choices:
   • Data Warehouse
   • Transaction Processing

5. Choose a deployment type.
   • Shared Infrastructure
     Run Autonomous Database on shared Exadata infrastructure.
   • Dedicated Infrastructure
     Run Autonomous Database on dedicated Exadata infrastructure.

Select Shared Infrastructure to create your instance on shared Exadata infrastructure.

See Create an Autonomous Data Warehouse Database on Dedicated Exadata Infrastructure for steps to create your instance on dedicated Exadata infrastructure.

6. Configure the database.
   • Choose database version
     Select a database version from the available versions.
   • OCPU Count
     Specify the number of CPU cores for your Autonomous Data Warehouse database.
   • Storage (TB)
     Specify the storage you wish to make available to your Autonomous Data Warehouse database, in terabytes.
   • Auto Scaling
     Deselect to disable auto scaling. By default auto scaling is enabled to allow the system to automatically use up to three times more CPU and IO resources to meet workload demand. See Use Auto Scaling for more information.

7. When a preview version is available, if you want to use the preview version, select Enable Preview Mode.
   If you want to use the preview version, also accept the preview termination period notice. See Preview Versions for Autonomous Database for more information.

   Note:
   A preview version is only available when there is an upcoming major version to release. At times no preview version will be available.

8. Create administrator credentials. Set the password for the Autonomous Data Warehouse 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 with Autonomous Data Warehouse.
   • Username
     This is a read only field.
• **Password** Set the password for the Autonomous Data Warehouse 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 with Autonomous Data Warehouse](#).

• **Confirm password** Enter the same password again to confirm your new password.

9. Choose network access

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The network access option you select, either <strong>Allow secure access from everywhere</strong> or <strong>Virtual cloud network</strong> cannot be changed after you provision your Autonomous Database (except by cloning a new database).</td>
</tr>
</tbody>
</table>

• **Allow secure access from everywhere**

By default all secure connections are allowed from everywhere. To restrict access configure an access control list (ACL). To add an ACL for the Autonomous Database, select **Configure access control rules**.

See [Configure an Access Control List with Autonomous Database](#) 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 Data Warehouse database.

See [Configure Private Endpoints with Autonomous Database](#) for more information.

10. Choose a license type

• **Bring Your Own License**

My organization already owns Oracle database software licenses. Bring my existing database software licenses to the database cloud service ([details](#)).

• **License Included**

Subscribe to new database software licenses and the database cloud service. (Optional) Click **Show Advanced Options** to enter additional options.

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.

11. Click **Create Autonomous Database**.

On the Oracle Cloud Infrastructure console the **State** shows Provisioning... until the new Autonomous Data Warehouse database is available.
Start Autonomous Data Warehouse

Describes the steps to start an Autonomous Data Warehouse instance.

- Sign in to your Oracle Cloud Account at cloud.oracle.com.
- From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
- On the Autonomous Databases page select an Autonomous Data Warehouse instance 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.

Note:

When an Autonomous Data Warehouse instance is started, Autonomous Data Warehouse CPU billing is initiated, billed by the second with a minimum usage period of one minute.

Stop Autonomous Data Warehouse

Describes the steps stop an Autonomous Data Warehouse instance.

- Sign in to your Oracle Cloud Account at cloud.oracle.com.
- From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
- On the Autonomous Databases page select an Autonomous Data Warehouse instance 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.

Note:

When an Autonomous Data Warehouse instance is stopped, the following details apply:

- Tools are no longer able to connect to a stopped instance.
- Autonomous Data Warehouse in-flight transactions and queries are stopped.
- Autonomous Data Warehouse CPU billing is halted.
Restart Autonomous Data Warehouse

Describes the steps to restart an Autonomous Data Warehouse instance.

- Sign in to your Oracle Cloud Account at cloud.oracle.com.
- From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
- On the Autonomous Databases page select an Autonomous Data Warehouse instance 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.

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.

Terminate Autonomous Data Warehouse

Describes the steps to terminate an Autonomous Data Warehouse instance.

Note:

Terminating an Autonomous Data Warehouse database permanently deletes the instance and removes all automatic backups. You cannot recover a terminated database.

1. Sign in to your Oracle Cloud Account at cloud.oracle.com.
2. From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
3. On the Autonomous Databases page select an Autonomous Data Warehouse instance 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 Database page enter the database name to confirm that you want to terminate the database.
3. Click Terminate Database.
Add CPU or Storage Resources or Enable Auto Scaling

Describes how to scale your Autonomous Data Warehouse on demand by adding CPU cores or storage (TB). Also describes how to enable auto scaling.

- Sign in to your Oracle Cloud Account at cloud.oracle.com.
- Open the Oracle Cloud Infrastructure console by clicking the next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
- On the Autonomous Databases page select an Autonomous Data Warehouse instance from the links under the Display Name column.

1. From the Details page click Scale Up/Down.

2. On the Scale Up/Down prompt, select the change in resources for your scale request.
   - Click up arrow to select a value for CPU Core Count. The default is no change.
   - Click up arrow to select a value for Storage (TB). The default is no change.
3. To enable auto scaling, select **Auto Scaling**.

Select 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 auto scaling disabled. The additional resources are available until you disable auto scaling by deselecting **Auto Scaling**. See [Use Auto Scaling](#) for more information.

4. Click **Update** to change your resources.

When you click **Update** with a resource change for CPU Core Count, Storage, or to enable or disable Auto Scaling, the Lifecycle State changes to **Scaling in Progress**. After the Lifecycle State changes to **Available** the changes apply immediately.

**Note:**

If auto scaling is disabled while more CPUs are in use than the specified CPU Core Count, then Autonomous Data Warehouse scales the number of CPU cores in use down to the CPU Core Count number.

---

**Remove CPU or Storage Resources or Disable Auto Scaling**

Describes how to scale your Autonomous Data Warehouse on demand by removing CPU cores or storage (TB). Also describes how to disable auto scaling.

- Sign in to your Oracle Cloud Account at [cloud.oracle.com](http://cloud.oracle.com).
- From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
- On the Autonomous Databases page select an Autonomous Data Warehouse instance from the links under the **Display Name** column.

1. From the **Details** page click **Scale Up/Down**.

2. On the **Scale Up/Down** page, select the change in resources for your scale request:
   - Click down arrow to select a value for **CPU Core Count**. The default is no change.
• Click down arrow to select a value for **Storage (TB)**. The default is no change.

3. When auto scaling is enabled deselect **Auto Scaling** to disable auto scaling.

4. Click **Update** to change your resources.

If auto scaling is disabled while more CPUs are in use than the specified CPU Core Count, then Autonomous Data Warehouse scales the number of CPU cores in use down to the CPU Core Count number.

---

**Use Auto Scaling**

Auto scaling is enabled by default when you create an Autonomous Database instance or you can use **Scale Up/Down** on the Oracle Cloud Infrastructure console to enable or disable auto scaling.

With auto scaling enabled the database can use up to three times more CPU and IO resources than specified by the number of OCPUs currently shown in the **Scale Up/Down** dialog. 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.

To see the average number of OCPUs used during an hour you can use the "Number of OCPUs allocated" graph on the Overview page on the Autonomous Data Warehouse service console. See **Console Overview** for more information.

Enabling auto scaling does not change the concurrency and parallelism settings for the predefined services. See **Managing Concurrency and Priorities on Autonomous Data Warehouse** for more information.

See **Add CPU or Storage Resources or Enable Auto Scaling** for the steps to enable auto scaling.
Managing Users on Autonomous Data Warehouse

This section describes administration tasks for managing users on Autonomous Data Warehouse.

Topics

- Create Users with Autonomous Data Warehouse
- Remove Users with Autonomous Data Warehouse
- Manage the Administrator Account on Autonomous Data Warehouse
- Manage User Privileges with Autonomous Data Warehouse
- Create and Update User Accounts for Oracle Machine Learning
- Use Microsoft Active Directory with Autonomous Database

Create Users with Autonomous Data Warehouse

To create users in your database, connect to the database as the ADMIN user using any SQL client tool.

For example, connect using Oracle SQL Developer (see Connect with Oracle SQL Developer (earlier than Version 18.2)).

- As the ADMIN user run the following SQL statement:

  ```sql
  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.

  This creates new_user with connect privileges. This user can now connect to Autonomous Data Warehouse and run queries. To grant additional privileges to users, see Manage User Privileges with Autonomous Data Warehouse.

  **Note:**

  The administrator needs to provide the credentials wallet to the user new_user. See Connecting to Autonomous Data Warehouse.
Autonomous Data Warehouse requires strong passwords; the password you specify must meet the default password complexity rules.

- 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.

Autonomous Data Warehouse requires strong passwords and uses the following password parameter values:

<table>
<thead>
<tr>
<th>Password Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASSWORD_GRACE_TIME</td>
<td>The number of days after the grace period begins during which a warning is issued and login is allowed.</td>
<td>7</td>
</tr>
<tr>
<td>PASSWORD_LIFE_TIME</td>
<td>The number of days the same password can be used for authentication.</td>
<td>360</td>
</tr>
<tr>
<td>PASSWORD_LOCK_TIME</td>
<td>The number of days an account will be locked after the specified number of consecutive failed login attempts</td>
<td>1</td>
</tr>
<tr>
<td>PASSWORD_REUSE_MAX</td>
<td>The number of password changes required before the current password can be reused.</td>
<td>4</td>
</tr>
<tr>
<td>PASSWORD_REUSE_TIME</td>
<td>The number of days before which a password cannot be reused.</td>
<td>1</td>
</tr>
</tbody>
</table>

See CREATE PROFILE in *SQL Language Reference* for more information on password parameters.

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;
```

For more information about the `ALTER USER` command, see *SQL Language Reference*.

See [Provide SQL Developer Web Access to Database Users](#) to add users for SQL Developer Web.

See [Create Oracle Application Express Workspaces in Autonomous Data Warehouse](#) for information on creating APEX workspaces.

See [Create and Update User Accounts for Oracle Machine Learning](#) to add user accounts for Oracle Machine Learning.
Remove Users with Autonomous Data Warehouse

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 (see Connect with Oracle SQL Developer (earlier than Version 18.2)).

- As the ADMIN user run the following SQL statement:

  ```sql
  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.

Manage the Administrator Account on Autonomous Data Warehouse

You can change the administrator user password and when locked unlock the administrator user account on Autonomous Data Warehouse.

**Topics**

- Change the Administrator Password in Autonomous Data Warehouse
- Unlock the Administrator Account in Autonomous Data Warehouse

Change the Administrator Password in Autonomous Data Warehouse

From the service 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 **Admin Password**.
2. On the **Admin Password** page enter the new password and confirm.
3. Click **Update**.

The password for the default administrator account, ADMIN, has the same password complexity rules mentioned in the section Create Users with Autonomous Data Warehouse.
Unlock the Administrator Account in Autonomous Data Warehouse

If you need to change the ADMIN user password, you can do this from the More Actions drop-down list, by selecting Admin Password.

Use the following steps to change the ADMIN password:

- Sign in to your Oracle Cloud Account at cloud.oracle.com.
- From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
- On the Autonomous Databases page select an Autonomous Data Warehouse instance from the links under the Display Name column.

See Signing in to Your Cloud Account in Getting Started with Oracle Cloud.

1. On the Details page, from the More Actions drop-down list, select Admin Password.

2. On the Admin Password dialog box enter the new password and confirm.

3. Click Update.

This operation also unlocks the ADMIN account if it was locked.
Manage User Privileges with Autonomous Data Warehouse

Autonomous Data Warehouse databases come with a predefined database role named **DWROLE**. This role provides the common privileges for the data warehouse developer or for a data scientist to perform real-time analytics. 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 with Oracle SQL Developer (18.2 or later)]).

2. As the ADMIN user grant **DWROLE**. For example, the following command grants **DWROLE** to the user **ADBUSER**:

   ```sql
   GRANT DWROLE TO adbuser;
   ```

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:
   - Use **DBMS_CLOUD_ADMIN.GRANT_TABLESPACE_QUOTA** to explicitly grant a quota to a user. See [GRANT_TABLESPACE_QUOTA Procedure](#) for more information.
   - Grant **UNLIMITED TABLESPACE** privileges to a user. For example, the following command grants unlimited tablespace privileges to the user **ADBUSER**:

     ```sql
     GRANT UNLIMITED TABLESPACE TO adbuser;
     ```

This privilege overrides any quota that was granted using **DBMS_CLOUD_ADMIN.GRANT_TABLESPACE_QUOTA**.

**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

---

Note:

The password for the default administrator account, ADMIN, has the same password complexity rules mentioned in the section [Create Users with Autonomous Data Warehouse](#).
Create and Update User Accounts for Oracle Machine Learning

An administrator can add an existing database user account to Oracle Machine Learning or create a new user account and user credentials with the Oracle Machine Learning User Management interface.

Topics:

• Create User
• Add Existing Database User Account to Oracle Machine Learning

Create User

An administrator creates a new user account 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. Select an Autonomous Data Warehouse instance and on the details page click Service Console.
2. On the Service Console click Administration.
3. Click Manage OML Users to open the Oracle Machine Learning User Administration page.
5. In the **Username** field, enter a username for the account. Using the username, the user will log in to an Oracle Machine Learning instance.

6. Enter a name in the **First Name** field.

7. Enter a name in the **Last Name** field.

8. In the **Email Address** field, enter the email ID of the user.

9. Select the option **Generate password and email account details to user. User will be required to reset the password on first sign in.** to auto generate a temporary password and send an email with the account credentials to the user. If you select this option, you need not enter values in the **Password** and **Confirm Password** fields; the fields are grayed out.

10. In the **Password** field, enter a password for the user, if you choose to create a password for the user. This option is disabled if you select the **Generate password...** option to auto generate a temporary password for the user.

11. In the **Confirm Password** field, enter a password to confirm the value that you entered in the **Password** field. By doing so, you create the password for the user. The user can change the password when first logging in.

12. Click **Create**.

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

An administrator adds an existing database user account 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 add an existing database user account:

1. Select an Autonomous Database instance, and on the details page click **Service Console**.
2. On the Service Console, click **Administration**.
3. Click **Manage OML Users** to add Oracle Machine Learning users.
4. 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.

5. 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.
6. Enter a name in the **First Name** field. (Optional)
7. Enter the last name of the user in the **Last Name** field. (Optional)
8. 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 a Oracle Machine Learning user.
9. 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 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 an Autonomous Data Warehouse database using their Active Directory credentials.

**Topics:**
- Configure CMU with Microsoft Active Directory on Autonomous Database
- Add Microsoft Active Directory Roles on Autonomous Database
- Add Microsoft Active Directory Users on Autonomous Database
- Connect to Autonomous Database with Active Directory User Credentials
- Tools Restrictions with Active Directory on Autonomous Database
- Verify Active Directory User Connection Information with Autonomous Database
- Remove Active Directory Users and Roles on Autonomous Database
- Disable Active Directory Access on Autonomous Database

**Configure CMU with Microsoft Active Directory on Autonomous Database**

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 users and roles to Microsoft Active Directory users and groups.

> **Note:**
CMU supports Microsoft Active Directory servers but does not support the Azure Active Directory service.

See Configuring Centrally Managed Users with Microsoft Active Directory for information on Centrally Managed Users (CMU).

The following are required before you configure the connection from Autonomous Database to Active Directory:

- You must have Microsoft Active Directory installed and configured. See the following for more information:
  - How To Configure Authentication For The Centrally Managed Users In An 18c Database (Doc ID 2462012.1)
  - AD DS Getting Started
• 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. Only password authentication is supported with CMU for Autonomous Database, so 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.

• The Active Directory servers must be accessible from Autonomous Database through the public internet and the port 636 of the Active Directory servers must be open to Autonomous Database in Oracle Cloud Infrastructure, so that Autonomous Database can have secured LDAP access over TLS/SSL to the Active Directory servers through the 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. Then you can use these RODCs in Oracle Cloud Infrastructure to authenticate and authorize the on-premise Active Directory users for access to Autonomous Databases. See Microsoft Windows: Extending Active Directory to Oracle Cloud Infrastructure for more information.

• You need to create or obtain the CMU configuration database wallet, cwallet.sso, and the CMU configuration file dsi.ora. These are the files you need to obtain and copy to configure your Autonomous Database to use Active Directory. You can configure CMU for an on-premise database and validate the wallet and the dsi.ora by 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.

The following limitation applies for CMU Active Directory with Autonomous Database:

• Only "password authentication" is supported for CMU with Autonomous Database. Other CMU authentication methods including Kerberos and PKI are not supported with Autonomous Database.

**Note:**

When you perform the configuration steps, connect to the Autonomous Data Warehouse database as the ADMIN user.

To configure CMU on Autonomous Database, as the ADMIN user, you upload the CMU configuration database wallet, cwallet.sso, and the CMU configuration file, dsi.ora, to your Object Store, and then use DBMS_CLOUD.GET_OBJECT to copy these files to a file system directory path specified by a directory object on the Autonomous Data Warehouse database.

To configure Autonomous Database to connect to Active Directory servers:
1. On your Autonomous Data Warehouse database, create a new directory object or choose an existing directory object. This is the directory where you store the wallet and the configuration file for connecting to Active Directory:

For example:

```sql
CREATE OR REPLACE DIRECTORY cmu_wallet_dir AS 'cmu_wallet';
```

2. Upload the CMU configuration database wallet, 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. Using `DBMS_CLOUD.GET_OBJECT` copy the CMU configuration database wallet and CMU configuration file from your Object Store to the directory that you created or chose in Step 1.

For example, use `DBMS_CLOUD.GET_OBJECT` to copy the files from Object Store to `CMU_WALLET_DIR` as follows:

```sql
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 => 'CMU_WALLET_DIR');
    DBMS_CLOUD.GET_OBJECT(
        credential_name => 'DEF_CRED_NAME',
        object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/dsi.ora',
        directory_name => 'CMU_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.

See `GET_OBJECT Procedure` for more information.

4. Set the Autonomous Data Warehouse database property `CMU_WALLET` to the name of the directory object that you created or chose in Step 1.

```sql
ALTER DATABASE PROPERTY SET CMU_WALLET='directory_object_name';
```

For example:

```sql
ALTER DATABASE PROPERTY SET CMU_WALLET='CMU_WALLET_DIR';
```
Use the following SQL statement to query the property value of the database property `CMU_WALLET`:

```sql
SELECT PROPERTY_VALUE FROM DATABASE_PROPERTIES WHERE PROPERTY_NAME='CMU_WALLET';
```

For example:

```
SQL> SELECT PROPERTY_VALUE FROM DATABASE_PROPERTIES WHERE PROPERTY_NAME='CMU_WALLET';

PROPERTY_VALUE
--------------------------------
CMU_WALLET_DIR
```

Use the following SQL statement to query the directory path of the directory object set to the property `CMU_WALLET`:

```sql
SELECT DIRECTORY_PATH FROM DBA_DIRECTORIES WHERE DIRECTORY_NAME='directory_object_name';
```

For example:

```
SQL> SELECT DIRECTORY_PATH FROM DBA_DIRECTORIES WHERE DIRECTORY_NAME='CMU_WALLET_DIR';

DIRECTORY_PATH
---------------------------------------------------------------
--------
/file_system_directory_path_example/cmucmu_wallet
```

In the `CREATE OR REPLACE DIRECTORY` statement in Step 1, if you want to preserve case for the directory object name then you need to include double quotes. For example:

```
CREATE OR REPLACE DIRECTORY "CMU_wallet_dir" AS 'cmu_wallet';
```

Then the case would be preserved and you would set the property `CMU_WALLET` as follows:

```
ALTER DATABASE PROPERTY SET CMU_WALLET='CMU_wallet_dir';
```

5. To maintain security remove the CMU configuration database wallet, `cwallet.sso` and the CMU configuration file `dsi.ora` from Object Store. For example, on Autonomous Database use `DBMS_CLOUD.DELETE_OBJECT` to delete these files from Object Store.

6. Set the `LDAP_DIRECTORY_ACCESS` parameter value to `PASSWORD` to enable the access from Autonomous Database to Active Directory.

```
ALTER SYSTEM SET LDAP_DIRECTORY_ACCESS=PASSWORD;
```
Add Microsoft Active Directory Roles on Autonomous Database

To add Active Directory roles on an Autonomous Data Warehouse database, map the database global roles to Active Directory groups with `CREATE ROLE` or `ALTER ROLE` statements (with `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 instance 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:

   ```sql
   CREATE ROLE global_role IDENTIFIED GLOBALLY AS
   'DN_of_an_AD_GROUP_of_WHICH_the_AD_USER_IS_a_MEMBER';
   ```

   For example:

   ```sql
   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:

   ```sql
   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 with Autonomous Data Warehouse for information on setting common privileges for Autonomous Database users.
4. If you want to add additional roles, follow these steps for each role.

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 an Autonomous Data Warehouse 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 instance 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 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 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, follow these steps for each Active Directory group.

See Configuring Authorization for Centrally Managed Users for more information on configuring users with Microsoft Active Directory.

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 Autonomous Data Warehouse 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.

• To log in to the Autonomous Data Warehouse 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 Autonomous Data Warehouse database using any of the connection methods described in Connecting to Autonomous Data Warehouse. 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.
Tools Restrictions with Active Directory on Autonomous Database

Note:

- Oracle Application Express is not supported for Active Directory users with Autonomous Database. See Create Oracle Application Express Workspaces in Autonomous Data Warehouse for information on using regular database users with Autonomous Database.
- Oracle SQL Developer Web is not supported for Active Directory users with Autonomous Database. See Provide SQL Developer Web Access to Database Users for information on using regular database users with Autonomous Database.
- Oracle Machine Learning is not supported for Active Directory users with Autonomous Database. See Add Existing Database User Account to Oracle Machine Learning for information on using regular database users with Autonomous Database.

Verify Active Directory User Connection Information with Autonomous Database

When users log in to the Autonomous Data Warehouse 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 Autonomous Data Warehouse 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 an Autonomous Data Warehouse database use standard database commands. This does not remove the related Active Directory users or groups that were mapped from the dropped database user or role.

To remove users from Autonomous Database:

1. Log in to the database instance 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.

See Remove Users with Autonomous Data Warehouse for more information.

### Disable Active Directory Access on Autonomous Database

Describes the steps to disable the LDAP access from Autonomous Data Warehouse database to Active Directory, and covers the steps to remove the CMU configuration from your Autonomous Data Warehouse database.

After you configure your Autonomous Data Warehouse database to access Active Directory, you can disable the access as follows:

1. Set the `LDAP_DIRECTORY_ACCESS` parameter value to `NONE` to disable the access from Autonomous Database to Active Directory:

```
ALTER SYSTEM SET LDAP_DIRECTORY_ACCESS=None;
```
2. You can re-enable the Active Directory access by setting the parameter value to PASSWORD:

```
ALTER SYSTEM SET LDAP_DIRECTORY_ACCESS=PASSWORD;
```

**Note:**

If you no longer want to maintain the connection to Active Directory, then do the following:

- Remove the database property `CMU_WALLET` by executing the following SQL statement:

```
ALTER DATABASE PROPERTY REMOVE CMU_WALLET;
```

- Remove the CMU configuration files, the database wallet `cwallet.sso` and `dsi.ora`, from the directory that you created or chose when you configured CMU. For example, use `DBMS_CLOUD.DELETE_FILE` to remove these files from `CMU_WALLET_DIR`.

See `DELETE_FILE Procedure` for more information on `DBMS_CLOUD.DELETE_FILE`. 
Managing and Monitoring Performance of Autonomous Data Warehouse

This section describes managing and monitoring the performance of Autonomous Data Warehouse on shared Exadata infrastructure.

For equivalent information about managing and monitoring performance of Autonomous Database on dedicated Exadata infrastructure, see these resources:

- Autonomous Data Warehouse on dedicated Exadata infrastructure: see Managing and Monitoring Performance.
- Autonomous Transaction Processing on dedicated Exadata infrastructure: see Managing and Monitoring Performance.

Topics

- Monitor the Performance of Autonomous Data Warehouse
- Managing Concurrency and Priorities on Autonomous Data Warehouse
- Manage CPU/IO Shares on Autonomous Data Warehouse
- Manage Runaway SQL Statements on Autonomous Data Warehouse
- Manage Optimizer Statistics on Autonomous Data Warehouse
- Manage Automatic Indexing on Autonomous Database
- Monitor Autonomous Data Warehouse with Performance Hub
- Monitor the Performance of Autonomous Data Warehouse with Oracle Management Cloud
- Monitor Performance with Autonomous Database Metrics

Monitor the Performance of Autonomous Data Warehouse

The **Overview** and **Activity** tabs in the Service Console provide information about the performance of the service. The **Activity** tab also shows past and current monitored SQL statements and detailed information about each statement.

Topics

- Monitor Activity and Utilization
- Console Overview
- Console Activity
- Monitor SQL Statements
Monitor Activity and Utilization

The **Overview** and **Activity** tabs show real-time and historical information about the utilization of the service.

Sign in to your Oracle Cloud Account at [cloud.oracle.com](http://cloud.oracle.com).

From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.

See Signing in to Your Cloud Account in *Getting Started with Oracle Cloud*.

The **Overview** tab shows general information about the utilization of the service.

1. From the Autonomous Data Warehouse landing pad select an instance.
2. From the **Details** page click **Service Console**.

3. Autonomous Data Warehouse shows the service console.
Note:

You can bookmark the service console URL and go to that URL directly without logging in to the Oracle Cloud Infrastructure console. If you logout and use the bookmark, then to see the service console you need to enter the ADMIN username, the password, and click Sign in. See Change the Administrator Password in Autonomous Data Warehouse if you need to change the password for the ADMIN user.

Console Overview

The Overview page shows real-time and historical information about the utilization of the service.

The components on this page are:

- **Storage**
  This chart shows the total and used storage capacity of the service. It indicates what percentage of the space is currently in-use. The used storage capacity is the total allocated space for all tablespaces.

- **CPU utilization (%)**
  This chart shows the historical CPU utilization of the service:
  - **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 CPUs the database is allowed to use which is two times the number of OCPUs. For example, if the database has four (4) OCPUs, the percentage in this graph is based on 8 CPUs.
Auto Scaling Enabled: For databases with auto scaling enabled the utilization percentage is reported with respect to the maximum number of CPUs the database is allowed to use, which is six times the number of OCPUs. For example, if the database has four OCPUs with auto scaling enabled the percentage in this graph is based on 24 CPUs.

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 OCPUs allocated
This chart shows the number of OCPUs allocated:

- Auto Scaling Disabled: For databases with 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.
- Auto Scaling Enabled: For databases with 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.

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.

The default retention period for performance data is eight days. So, the CPU utilization, running statements, and average SQL response time charts show data for the last eight days by default.

Note:
The retention time can be changed by modifying the Automatic Workload Repository retention setting with the PL/SQL procedure DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS. Be aware that increasing the retention time will result in more storage usage for performance data. See Oracle® Database PL/SQL Packages and Types Reference.

Console Activity
The Activity page shows real-time and historical information about the utilization of the service.

To access detailed information about the service performance click the Activity tab in the service console.
Note:
The default view in this tab is real-time. This view shows performance data for the last hour.

The components 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**
  This chart shows the CPU utilization of each consumer group. The utilization percentage is reported with respect to the number of CPUs the database is allowed to use which is two times the number of OCPUs. For example, if the database has four (4) OCPUs, the percentage in this graph is based on 8 CPUs.
  For databases with auto scaling enabled the utilization percentage is reported with respect to the maximum number of CPUs the database is allowed to use, which is six times the number of OCPUs. For example, if the database has four OCPUs with auto scaling enabled the percentage in this graph is based on 24 CPUs.
  See *Managing Concurrency and Priorities on Autonomous Data Warehouse* for detailed information on consumer groups.

- **Running Statements**
  This chart shows the average number of running SQL statements in each consumer group. See *Managing Concurrency and Priorities on Autonomous Data Warehouse* for detailed information about consumer groups.

- **Queued Statements**
  This chart shows the average number of queued SQL statements in each consumer group. See *Managing Concurrency and Priorities on Autonomous Data Warehouse* for detailed information on consumer groups.
To see earlier data click **Time period**. The default retention period for performance data is eight days. So, this view shows information for the last eight days by default.

**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*. 
In the time period view you can use the calendar to look at a specific time in the past eight days.
You can also use the time slider to change the period for which performance data is shown.

Monitor SQL Statements

The Monitored SQL tab shows information about current and past monitored SQL statements.

See About Monitoring Database Operations in Database SQL Tuning Guide.

Click the Monitored SQL tab to see these statements.

To see the detailed SQL Monitor report for a statement, select a statement and click Show Details. The Overview tab in the pop-up shows general information for that statement.
Click **Plan Statistics** tab to see the runtime execution plan of the statement.

Click **Parallel** tab to see information about the parallel processes, if the statement uses parallelism.
If you want to download an active SQL Monitor report for a statement, select the statement in the Monitored SQL page and click Download report. This will save the active SQL Monitor report to your client. See About Monitoring Database Operations in Database SQL Tuning Guide.

To cancel a running statement, select that statement in the Monitored SQL list and click Cancel execution.

Managing Concurrency and Priorities on Autonomous Data Warehouse

Concurrency and prioritization of user requests in Autonomous Data Warehouse is determined by the database service the user is connected with.

Overview

Users are required to select a service when connecting to the database. The service names 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 provision an Autonomous Data Warehouse service with the name ADW1, your service names are:

- `adw1_high`
- `adw1_medium`
- `adw1_low`

For example, a user connecting with the `adw1_low` service uses the consumer group LOW.

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.
- **LOW**: Least resources, highest concurrency. Queries run serially.
Idle Time Limits

Autonomous Data Warehouse 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.

See MAX_IDLE_TIME for more information.

Concurrency

The concurrency level of these consumer groups changes based on the number of OCPUs you subscribe to. The HIGH consumer group’s concurrency is fixed and does not change based on the number of OCPUs. The MEDIUM and LOW consumer groups can run more concurrent SQL statements if you scale up the compute capacity of your service.

Note:

The HIGH consumer group is configured for low concurrency, even a single query in this consumer group can use all resources in your database. If your workload has concurrent queries Oracle recommends using the MEDIUM consumer group. If your concurrency requirements are not met with the MEDIUM consumer group, you can use the LOW consumer group or you can scale up your compute capacity and continue using the MEDIUM consumer group.

For example, for an Autonomous Data Warehouse with 16 OCPUs, 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 these concurrency levels are reached for a consumer group new SQL statements in that consumer group will be queued until one or more running statements finish.

Predefined Job Classes with Oracle Scheduler

Autonomous Data Warehouse includes predefined job_class values to use with Oracle Scheduler.

The predefined job_class values, LOW, MEDIUM, and HIGH 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 Data Warehouse.

For example: use the following to create a single regular job to run in `HIGH` consumer group:

```sql
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;
/
```

**Note:**
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.

See Scheduling Jobs with Oracle Scheduler for more information on Oracle Scheduler and `DBMS_SCHEDULER.CREATE_JOB`.

### Manage CPU/IO Shares on Autonomous Data Warehouse

Autonomous Data Warehouse 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.

By default, the CPU/IO shares assigned to the consumer groups `HIGH`, `MEDIUM`, `LOW` are 4, 2, and 1, respectively. The shares determine how much CPU/IO resources a consumer group can use with respect to the other consumer groups. 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.

You can set CPU/IO shares from the service console or using the PL/SQL package `cs_resource_manager.update_plan_directive`.

Follow these steps to set CPU/IO shares from the service console:

1. From the Autonomous Data Warehouse details page, click **Service Console**.
2. On the Service Console click **Administration**.
3. Click **Set Resource Management Rules**.
4. Select **CPU/IO shares** to set CPU/IO share values for consumer groups.
5. Set the CPU/IO share values.
6. Click **Save changes**.

For example, the following figure shows CPU/IO share values that you can modify:

![Set Resource Management Rules](image)

Click **Load Default Values** to load the default values; then click **Save changes** to apply the populated values.

You can also change the default values using the PL/SQL procedure `cs_resource_manager.update_plan_directive`:

For example, running the following script with the ADMIN user sets CPU/IO shares to 8, 2, and 1 for consumer groups HIGH, MEDIUM, and LOW respectively. This will allow 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:

```sql
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;
/
```
Manage Runaway SQL Statements on Autonomous Data Warehouse

Specifies how you configure Autonomous Data Warehouse to terminate SQL statements automatically based on their runtime or the amount of IO they are doing.

You can set rules from the service console or using the PL/SQL package `cs_resource_manager`.

Follow these steps to set rules from the service console:

1. From the Autonomous Database details page, click **Service Console**.
2. On the Service Console click **Administration**.
3. Click **Set Resource Management Rules**.
4. Select the **Run-away criteria** tab to set rules for consumer groups.
5. Select the **Consumer group**: **HIGH**, **MEDIUM**, or **LOW**.
6. Set runaway criteria values:
   - Query run time (seconds)
   - Amount of IO (MB)
7. Click **Save changes**.

For example, the following shows values for setting the runtime limit to 120 seconds and the IO limit to 1000MB for the **HIGH** consumer group:

![Set Resource Management Rules](image)

When a SQL statement in the specified consumer 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;
/
```

### Manage Optimizer Statistics on Autonomous Data Warehouse

Describes Autonomous Data Warehouse commands to run when you need to gather optimizer statistics or enable optimizer hints.

#### Managing Optimizer Statistics

Autonomous Data Warehouse 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.
Managing Optimizer Hints

Autonomous 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:

```sql
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:

```sql
ALTER SESSION
    SET OPTIMIZER_IGNORE_PARALLEL_HINTS=FALSE;
```

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:

   ```sql
   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:](image)

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):

   ```sql
   EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE', 'OFF');
   ```

See Managing Auto Indexes for more information.

## Monitor Autonomous Data Warehouse with Performance Hub

You can view real-time and historical performance data from the Performance Hub. Performance Hub shows Active Session History (ASH) analytics, SQL monitoring and workload information.

Sign in to your Oracle Cloud Account at [cloud.oracle.com](http://cloud.oracle.com).

From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.

1. From the Autonomous Data Warehouse landing pad select an instance.
2. From the Details page click Performance Hub.

The Performance Hub page is displayed. This page has two sections:

- The Time Range selector.
- The tabbed data area, with the tabs ASH Analytics, SQL Monitoring and Workload.

### The Time Range Selector

The Time Range selector is on the top of the Performance Hub page. Use the Select Duration field to set the time duration. By default, Last hour is selected. You can choose to view Last 8 hours, Last 24 hours, Last week, or specify a custom time range using the Custom option.

The Time Range field shows active sessions in chart form for the time period selected. The active sessions chart displays the average number of active sessions broken down by CPU, User I/O, and Wait. The active sessions chart also shows the Max CPU usage.
The sliding box on the time range chart is the **time slider**. Use the time slider to select the exact period of time for which data is displayed in the Performance Hub tables and graphs. This is a subsection of the period of time shown in the **Time Range** field.

You can slide the box to the left or the right to shift the time period under analysis. To slide the entire box, left-click anywhere inside the box and drag the box to the left or the right. Widen or narrow the box to increase or decrease the length of time under analysis. To widen or narrow the box, left-click and hold the handlebar on either side of the box, then drag to the left or the right to increase or decrease the size of the current time range box.

Click **Refresh** to refresh the data in Performance Hub according to the time range chosen.

**The ASH Analytics Tab**

This tab, which is displayed by default, shows Active Session History (ASH) analytics charts to explore 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.

See Active Session History (ASH) in *Oracle Database Concepts* for more information on Active Session History.

**The SQL Monitoring Tab**

The 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**.

The **Status** column has the following icons:

- A spinning icon indicates that the SQL statement is executing.
- A green check mark icon indicates that the SQL statement completed its execution during the specified time period.
- A red cross icon indicates that the SQL statement did not complete, either due to an error, or due to the session being terminated.
- A clock icon indicates that the SQL statement is queued.

To terminate a running or queued SQL statement, click **Kill Session**.

Select the link in the **SQL ID** column to go to the corresponding **Real-time SQL Monitoring** page. This page provides additional details to help you tune the selected SQL statement.

**The Workload Tab**

This tab shows four chart areas that show the workload on the database in various ways:

- **CPU Statistics**: Charts CPU usage as measured by the statistic you select:
  - **CPU Time**: Shows how many CPU seconds are being used per second by the database's foreground sessions.
– **CPU Utilization (%)**: Shows the CPU usage of all the database’s consumer groups as a percentage of the number of CPUs the database is allowed to use.

• **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 as measured by the statistic you select:
  – **User Calls and Transactions**: Shows the **User Calls**, **Executions** and **Transactions** statistics in a single, consolidated chart.
  – **User Calls**: Shows the number of user calls (such as login, parse, fetch, or execute) per second.
  – **Executions**: Shows the number of executed SQL statements per second, whether initiated directly by a user or recursively.
  – **Transactions**: Shows the combined number of user commits and user rollbacks per second.
  – **Parses**: Shows the combined number of hard and soft parses per second.
  – **Running Statements**: Shows the number of running SQL statements across all the database’s consumer groups.
  – **Queued Statements**: Shows the number of queued parallel SQL statements across all the database’s consumer groups.

• **Sessions**:
  – **Current Logons**: Shows the number of current successful logons.
  – **Sessions**: Shows the number of sessions.

### Monitor the Performance of Autonomous Data Warehouse with Oracle Management Cloud

Oracle Management Cloud allows you to monitor your Autonomous Data Warehouse database availability and performance. 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 Data Warehouse 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 Autonomous Data Warehouse databases with metrics, alarms, and notifications. You can use Oracle Cloud Infrastructure console or Monitoring APIs to view metrics.

**Topics**
View Metrics for an Autonomous Database Instance

Shows the steps to view the Autonomous Database metrics.

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.

To view metrics for an Autonomous Database instance:

• Sign in to your Oracle Cloud Account at cloud.oracle.com.
• From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
• On the Autonomous Databases page select an Autonomous Data Warehouse instance from the links under the Display Name column.
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.

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 Database Metrics.

You can also use the Monitoring API to view metrics. See Monitoring API 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.

To use the metrics service to view Autonomous Database metrics:

• Sign in to your Oracle Cloud Account at cloud.oracle.com.
• From the left navigation list click Monitoring > Service Metrics.
1. On the Service Metrics page, under Compartment select your compartment.
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.

See **Database Metrics** for a list of the database metrics and dimensions.
Back Up and Restore of Autonomous Data Warehouse

This section describes backup and recovery tasks on Autonomous Data Warehouse.

Topics

• About Backup and Recovery on Autonomous Data Warehouse
• Restore and Recover your Autonomous Data Warehouse Database
• Manual Backups on Autonomous Data Warehouse

About Backup and Recovery on Autonomous Data Warehouse

Autonomous Data Warehouse 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.

Manual Backups

You do not have to do any manual backups for your database as Autonomous Data Warehouse backs up your database automatically. You can do manual backups using the cloud console; for example if you want to take a backup before a major change to make restore and recovery faster. The manual backups are put in your Cloud Object Storage bucket. When you initiate a point-in-time recovery Autonomous Data Warehouse decides which backup to use for faster recovery.

Recovery

You can initiate recovery for your Autonomous Data Warehouse database using the cloud console. Autonomous Data Warehouse automatically restores and recovers your database to the point-in-time you specify.

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.
### About Backup and Recovery on Autonomous Data Warehouse

**Sales Autonomous Data Warehouse**

- **Database Name:** DB20
- **Workload Type:** Data Warehouse
- **Compartment:** (root)
- **OCID:** _oqv2qgShow Copy
- **Created:** Wed, Dec 12, 2018, 18:00:36 UTC
- **CPU Count:** 1
- **Storage (TB):** 1
- **License Type:** Bring Your Own License (BYOL)
- **Database Version:** 18c
- **Auto Scaling:** Enabled
- **Lifecycle State:** Available
- **Instance Type:** Paid

#### Infrastructure
- **Dedicated Infrastructure:** No

#### Backup
- **Last Automatic Backup:** Mon, Feb 24, 2020, 09:26:23 UTC

#### Network
- **Access Type:** Allow secure access from anywhere
- **Access Control List:** Disabled

#### Maintenance
- **Next Maintenance:** Sun, Mar 1, 2020, 24:00:00 UTC
- **Maintenance:** 04:00:00 UTC

### Backups

Backups are automatically created daily.

```
<table>
<thead>
<tr>
<th>Display Name</th>
<th>State</th>
<th>Type</th>
<th>Started</th>
<th>Ended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb 24, 2020</td>
<td>Active</td>
<td>Incremental, initiated by Auto Backup</td>
<td>Mon, Feb 24, 2020, 09:23:44 UTC</td>
<td>Mon, Feb 24, 2020, 09:26:23 UTC</td>
</tr>
<tr>
<td>Feb 23, 2020</td>
<td>Active</td>
<td>Incremental, initiated by Auto Backup</td>
<td>Sun, Feb 23, 2020, 09:30:28 UTC</td>
<td>Sun, Feb 23, 2020, 09:45:58 UTC</td>
</tr>
<tr>
<td>Feb 22, 2020</td>
<td>Active</td>
<td>Incremental, initiated by Auto Backup</td>
<td>Sat, Feb 22, 2020, 14:02:50 UTC</td>
<td>Sat, Feb 22, 2020, 14:16:20 UTC</td>
</tr>
<tr>
<td>Feb 21, 2020</td>
<td>Active</td>
<td>Incremental, initiated by Auto Backup</td>
<td>Fri, Feb 21, 2020, 09:24:01 UTC</td>
<td>Fri, Feb 21, 2020, 09:27:16 UTC</td>
</tr>
<tr>
<td>Feb 20, 2020</td>
<td>Active</td>
<td>Incremental, initiated by Auto Backup</td>
<td>Thu, Feb 20, 2020, 09:22:20 UTC</td>
<td>Thu, Feb 20, 2020, 09:25:20 UTC</td>
</tr>
</tbody>
</table>
```
Backup Note - Files on Object Store

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 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 Data Warehouse Database for information on Restore.

See Clone Autonomous Data Warehouse from a Backup for information on Clone from a backup.

Restore and Recover your Autonomous Data Warehouse Database

The Autonomous Data Warehouse console More Actions drop-down Restore operation allows you to initiate recovery for your database.

To restore and recover your database to a point in time, do the following:

• Sign in to your Oracle Cloud Account at cloud.oracle.com.

• From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.

• On the Autonomous Databases page select an Autonomous Data Warehouse instance from the links under the Display Name column.

1. On the details page, from the More Actions drop-down list, select Restore to display the Restore prompt.
2. In the Restore prompt, select **Specify Timestamp** or **Select Backup** to restore to a point in time or to restore from a specified backup.

   - SPECIFY TIMESTAMP: Enter a timestamp to restore to in the ENTER TIMESTAMP calendar field.
• 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.

### Restore

![Restore Autonomus Database](image)

3. Click **Restore**.

   **Note:**

   Restoring Autonomous Data Warehouse 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.

   The details page shows **Lifecycle State**: Restore In Progress...

4. When the restore operation finishes your Autonomous Data Warehouse instance opens in read-only mode and the instance details page **Lifecycle State** shows Available Needs Attention.
5. At this point you can connect to your Autonomous Data Warehouse instance and check your data to validate that the restore point you specified was correct:

- If the restore point you specified was correct and you want to open your database in read-write mode click **Stop** and after the database stops, click **Start** to start the database. After stopping and starting, the database opens in read-write mode.

  **Note:**
  After the instance is opened in read-write mode some of your backups are invalidated.

- 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.
Manual Backups on Autonomous Data Warehouse

In addition to automatic backups, Autonomous Data Warehouse also allows you to take manual backups to your Oracle Cloud Infrastructure Object Storage.

Topics:

- Configure Manual Backups on Autonomous Data Warehouse
- Perform Manual Backups on Autonomous Data Warehouse
Configure Manual Backups on Autonomous Data Warehouse

Follow these steps to define your Oracle Cloud Infrastructure Object Storage credentials and tenancy URL and to create the bucket for manual backups.

To perform manual backups you need to define your Oracle Cloud Infrastructure Object Storage credentials and your Oracle Cloud Infrastructure Object Storage tenancy URL; you also need to create a bucket to hold the backups. The manual backup configuration tasks are a one-time operation. After you define your credentials and your tenancy URL you can initiate manual backups without doing the same operations again unless the URL, the credentials, or bucket change.

For this task you need a client tool that is connected to Autonomous Data Warehouse to perform the DDL commands. For example, use SQL Developer or SQL*Plus. See Connect to Autonomous Data Warehouse Using Oracle Database Tools.

1. Set the database default_bucket property to your Oracle Cloud Infrastructure Object Storage tenancy URL. The format of the tenancy URL is https://swiftobjectstorage.region.oraclecloud.com/v1/object_storage_namespace. Note that you need to do this using the ADMIN user.

For example:

```sql
ALTER DATABASE PROPERTY SET default_bucket='https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/namespace-string';
```

Where namespace-string is the Oracle Cloud Infrastructure object storage namespace. See Understanding Object Storage Namespaces for more information.

2. On your Oracle Cloud Infrastructure Object Storage, create a bucket to hold the backups. The format of the bucket name is backup_databasename. Where databasename is lowercase.

   a. Open the Oracle Cloud Infrastructure service console from MyServices through the Services menu or the Services Dashboard tile.
   b. Select Object Storage from the menu.
   c. Select Object Storage from the submenu.
   d. Create a bucket in a compartment by clicking Create Bucket.

Manual backups 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.

For example, if you provision an Autonomous Data Warehouse instance named ADWC1, the bucket name should be backup_adwc1. Following the same example, the URL of this bucket would be (the bucket name is lowercase):

```text
https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/namespace-string/backup_adwc1
```
Where `namespace-string` is the Oracle Cloud Infrastructure object storage namespace. See Understanding Object Storage Namespaces for more information.

3. Create a credential for your Oracle Cloud Infrastructure Object Storage account using `DBMS_CLOUD.CREATE_CREDENTIAL`. See CREATE_CREDENTIAL Procedure.

    Note that you need to do this using the ADMIN user. For example:

    ```sql
    BEGIN
        DBMS_CLOUD.CREATE_CREDENTIAL(
            credential_name => 'DEF_CRED_NAME',
            username => 'adwc_user@example.com',
            password => 'password'
        );
    END;
    /```

4. Set the database property `DEFAULT_CREDENTIAL` to the credential you created in the previous step. For example:

    ```sql
    ALTER DATABASE PROPERTY SET DEFAULT_CREDENTIAL = 'ADMIN.DEF_CRED_NAME';
    ```

    To list the current value for the default bucket, use the following command:

    ```sql
    SELECT PROPERTY_VALUE FROM DATABASE_PROPERTIES WHERE PROPERTY_NAME='DEFAULT_BUCKET';
    ```

**Perform Manual Backups on Autonomous Data Warehouse**

In addition to automatic backups Autonomous Data Warehouse also allows you take manual backups to your Oracle Cloud Infrastructure Object Storage.

- Sign in to your Oracle Cloud Account at [cloud.oracle.com](https://cloud.oracle.com).
- From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
- On the Autonomous Databases page select an Autonomous Data Warehouse instance from the links under the Display Name column.

1. On the details page, under Resources, click Backups.
2. On the details page, under Backups, click Create Manual Backup.
3. In the **Create Manual Backup** dialog enter a name in the **Name** field.

4. In the **Create Manual Backup** dialog click **Create**.

**Note 1:**

Each manual backup creates a full backup on your Oracle Cloud Infrastructure Object Storage bucket and the backup can only be used by the Autonomous Data Warehouse instance when you initiate a point-in-time-recovery.
Note 2:
The retention period for manual backups is the same as automatic backups which is 60 days.

Note 3:
While backing up a database, the database is fully functional; however during the backup lifecycle management operations are not allowed. For example, stopping the database is not allowed during the backup.
Cloning, Moving, or Upgrading an Autonomous Data Warehouse Database

Autonomous Data Warehouse provides cloning where you can choose to clone either the full database or only the database metadata. You can also use cloning to upgrade your database to a newer Oracle Database version if your region has multiple Oracle Database versions available (some regions only have one Oracle Database version). You can move an Autonomous Data Warehouse database to a different Oracle Cloud Infrastructure compartment.

When you clone an Autonomous Data Warehouse instance, 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.

When you perform the clone operation you can select a newer Oracle Database version for the target database. This allows you to use cloning to upgrade your Autonomous Data Warehouse database.

If your Autonomous Database instance is using Oracle Database 18c you can manually upgrade to Oracle Database 19c.

When an upcoming release is available for preview with Autonomous Database, you can clone your existing database or clone a backup, to use the preview version. A preview version is only available when there is an upcoming major version to release. At times no preview version will be available.

Topics

- Clone an Autonomous Data Warehouse Instance
- Clone Autonomous Data Warehouse from a Backup
- Notes for Cloning Autonomous Database
- Move an Autonomous Data Warehouse Database to a Different Compartment
- Upgrade an Autonomous Database Instance to Oracle Database 19c

Clone an Autonomous Data Warehouse Instance

Shows you the steps to clone an Autonomous Data Warehouse database from the Oracle Cloud Infrastructure Console.

- Sign in to your Oracle Cloud Account at cloud.oracle.com.
- Open the Oracle Cloud Infrastructure console by clicking the ☑️ next to Oracle Cloud.
From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.

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 Data Warehouse 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.
   - Metadata Clone: creates a new database with the source database’s metadata without the data.

6. In the Configure 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 using a selected backup or from a point-in-time timestamp that you enter. See Clone Autonomous Data Warehouse from a Backup for more information.

7. Provide basic information for the Autonomous Database.
   - 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 14 characters.

    **Note:**
    The same database name cannot be used for multiple Oracle Autonomous Database databases in the same tenancy in the same region.

8. Configure the database.
   - Choose database version Select a database version that is the same version or is a higher version than the source database.
   - OCPU Count Specify the number of CPU cores for your Autonomous Data Warehouse database.
   - Storage (TB) Specify the storage you wish to make available to your Autonomous Data Warehouse database, in terabytes.
For a **Full Clone**, the minimum storage that you can specify is the source database’s actual used space rounded to the next TB.

- **Auto scaling** Deselect to disable auto scaling. With auto scaling the system to automatically uses up to three times more CPU and IO resources to meet workload demand. See Use Auto Scaling for more information.

9. Create administrator credentials.

- **Username** This is a read-only field.
- **Password** Set the password for the Autonomous Data Warehouse 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 with Autonomous Data Warehouse.
- **Confirm password** Specify a value to confirm the password.

10. Choose network access

    **Note:**

    The network access option you select, either **Allow secure access from everywhere** or **Virtual cloud network** cannot be changed after you provision your Autonomous Database (except by cloning a new database).

- **Allow secure access from everywhere**

    By default all secure connections are allowed from everywhere. To restrict access configure an access control list (ACL). To add an ACL for the Autonomous Database, select **Configure access control rules**.

    See Configure an Access Control List with Autonomous Database 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 Data Warehouse database.

    See Configure Private Endpoints with Autonomous Database for more information.

11. Choose a license type.

- **Bring Your Own License**

    My organization already owns Oracle database software licenses. Bring my existing database software licenses to the database cloud service ([details](#)).

- **License Included**

    Subscribe to new database software licenses and the database cloud service.

    (Optional) Click **Show Advanced Options** to enter additional options.
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.

On the Oracle Cloud Infrastructure console the **State** shows Provisioning... until the new Autonomous Data Warehouse database is available.

See [Notes for Cloning Autonomous Database](#) for additional information on cloning.

See [Cloning an Autonomous Database](#) for information on using the API.

## Clone Autonomous Data Warehouse from a Backup

Shows the options to select a backup as the clone source for cloning Autonomous Database.

- Sign in to your Oracle Cloud Account at [cloud.oracle.com](http://cloud.oracle.com).
- Open the Oracle Cloud Infrastructure console by clicking the ⬇️ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.

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 Data Warehouse 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 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.
   - **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 Data Warehouse Instance](#) for details and steps with this selection.
   - **Clone from a backup**: This selection creates a database clone using a backup or lets you enter a point-in-time to create the clone. 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.

**Point in time clone**
Select the backup from a list

**Note:**

When creating a clone from a backup, you must select a backup that is at least 2 hours old, or the clone operation will fail.

8. Provide basic information for the Autonomous Database.
   - **Create in Compartment.** See [Compartment](#) 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 14 characters.
9. Configure the database.
   • **Choose database version** Select a database version that is either the same version or is a higher version than the source database.
   • **OCPUs Count** Specify the number of CPU cores for your Autonomous Data Warehouse database.
   • **Storage (TB)** Specify the storage you wish to make available to your Autonomous Data Warehouse database, in terabytes.
     For a **Full Clone**, the minimum storage that you can specify is the source database's actual used space rounded to the next TB.
   • **Auto scaling** Deselect to disable auto scaling. With auto scaling the system to automatically uses up to three times more CPU and IO resources to meet workload demand. See **Use Auto Scaling** for more information.

10. Create administrator credentials.
    • **Username** This is a read-only field.
    • **Password** Set the password for the Autonomous Data Warehouse 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 with Autonomous Data Warehouse**.
    • **Confirm password** Specify a value to confirm the password.

11. Choose network access

   **Note:**
   The network access option you select, either **Allow secure access from everywhere** or **Virtual cloud network**, cannot be changed after you provision your Autonomous Database (except by cloning a new database).

   • **Allow secure access from everywhere**
     By default all secure connections are allowed from everywhere. To restrict access configure an access control list (ACL). To add an ACL for the Autonomous Database, select **Configure access control rules**.
     See **Configure an Access Control List with Autonomous Database** 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 Data Warehouse database.

See Configure Private Endpoints with Autonomous Database for more information.

12. Choose a license type.

  - **Bring Your Own License**
    My organization already owns Oracle database software licenses. Bring my existing database software licenses to the database cloud service (details).

  - **License Included**
    Subscribe to new database software licenses and the database cloud service.

    (Optional) Click Show Advanced Options to enter additional options.

    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.

    On the Oracle Cloud Infrastructure console the **State** shows Provisioning... until the new Autonomous Data Warehouse 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).

- When creating a clone from a backup, you must select a backup that is at least 2 hours old, or the clone operation will fail.

- 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.

---

# Notes for Cloning Autonomous Database

Provides information about the cloning operation and the resulting cloned database when you clone an Autonomous Data Warehouse database.
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).

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.

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 with Autonomous Database for more information.

For the clone operation, note the following limitation:

- You can only clone an Autonomous Data Warehouse instance to the same tenancy and the same region as the source database.

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.
- Performance data for the time before the clone operation is not visible on the service console of the cloned database.

For more information on setting resource management rules, see Manage Runaway SQL Statements on Autonomous Data Warehouse.

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 clone clears the statistics for that table and updates the statistics with the new load.
For more information on Optimizer Statistics, see Optimizer Statistics Concepts.

Move an Autonomous Data Warehouse Database to a Different Compartment

Shows you the steps to move an Autonomous Data Warehouse database to a different Oracle Cloud Infrastructure compartment.

Note:

- To move an Autonomous Data Warehouse 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 an Autonomous Data Warehouse 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.

- Sign in to your Oracle Cloud Account at cloud.oracle.com.
- Open the Oracle Cloud Infrastructure console by clicking the next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
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 Data Warehouse instance from the list in your compartment.
5. In the Move Resource to a Different Compartment page, select the new compartment.
6. Click **Move Resource**.

See **Moving Database Resources to a Different Compartment** for more information.

**Upgrade an Autonomous Database Instance to Oracle Database 19c**

Depending on your region, Autonomous Database provides Oracle Database 18c and Oracle Database 19c as database versions.

If your Autonomous Database instance is on Oracle Database 18c, after notifying you of the upgrade date, Oracle will automatically upgrade your instance to Oracle Database 19c. If you want to upgrade at a time of your choosing, then you can manually upgrade your database to Oracle Database 19c. By choosing to manually upgrade, you can start using the Autonomous Database features that are available in Oracle Database 19c prior to the automatic upgrade date.

---

**Note:**

If your Autonomous Database instance is on Oracle Database 18c, Oracle recommends cloning the instance to Oracle Database 19c and testing your applications before doing the actual upgrade.

If you are using Always Free Autonomous Database, you can wait for your Always Free instance to be automatically upgraded to Oracle Database 19c. You can also manually upgrade to Oracle Database 19c by cloning your Always Free database to a paid instance.

The following sections cover the database version upgrade methods for Autonomous Database:

- Manually Upgrade to Oracle Database 19c
- Manually Upgrade by Performing a Full Clone
• **Manually Upgrade by Cloning from a Backup**

• **Automatically Upgrade to Oracle Database 19c**

See [Oracle Database Versions and Availability by Region](#) for the list of regions and database versions.

---

### Manually Upgrade to Oracle Database 19c

If your Autonomous Database instance is using Oracle Database 18c you can manually upgrade to Oracle Database 19c.

The manual upgrade option is only available when the database version of your Autonomous Database instance is Oracle Database 18c. You can view the database version on the Autonomous Database details page in the **Database Version** field.

#### Note:

To perform a manual upgrade, the Autonomous Database must be available (the **Lifecycle State** field must show **Available**).

To manually upgrade your Autonomous Database to Oracle Database 19c, do the following:
• Sign in to your Oracle Cloud Account at cloud.oracle.com.

• Open the Oracle Cloud Infrastructure console by clicking the ☰ next to Oracle Cloud.

• From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.

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 Data Warehouse instance from the list in your compartment.

4. On the Details page, in the Oracle Database 19c banner click Upgrade to 19c.

Alternatively, in the Database Version field you can click the link Upgrade to 19c.
Oracle Cloud Infrastructure shows the Upgrade Autonomous Database dialog.

5. In the Upgrade Autonomous Database dialog, enter the database name to confirm that you want to upgrade your database.

6. Click Upgrade.

After you click Upgrade, the Lifecycle State shows Upgrading and the upgrade proceeds as follows:
• To minimize downtime, when the upgrade is ongoing you can use the Oracle Database 18c database during most of the upgrade process. The Oracle Cloud Infrastructure console shows the following banner:

![Banner](image)
The Oracle Database software version is being upgraded. The database remains available during the upgrade.

When the Oracle Cloud Infrastructure console shows this banner the database is available and online.

• Next, the Oracle Cloud Infrastructure console shows the following banner:

![Banner](image)
The database is being restarted to complete a software upgrade. The database will be available after the restart completes.

When Oracle Cloud Infrastructure shows this banner the database is unavailable for up to five minutes while it is restarted.

• When the restart completes the upgraded database is available and the Database Version field shows 19c.

Manual upgrade notes:

• The upgraded Oracle Database 19c instance uses the same wallet as the Oracle Database 18c instance. You can continue using your existing wallets to connect to the database. See Download Client Credentials (Wallets) for more information.

• The manual upgrade option is not available with Always Free Autonomous Databases.

• There is a known issue that after you complete a manual upgrade, the list of backups only shows the backups that occur after the upgrade date. See About Backup and Recovery on Autonomous Data Warehouse for information on listing backups.

**Manually Upgrade by Performing a Full Clone**

If your Autonomous Database instance is using Oracle Database 18c you can upgrade to Oracle Database 19c by creating a full clone.

To upgrade your Autonomous Database by creating a full clone, do the following:

1. Verify the Oracle Database version for your Autonomous Database instance is Oracle Database 18c.

   On the Oracle Cloud Infrastructure console, the Autonomous Database details page includes the Database Version field showing your database version.

2. During the cloning operation, in the Clone Type area select Full Clone, and in the Clone source area, select Clone from database instance.

3. During the cloning operation, in the cloning step where you configure the database, select the newer Oracle Database version in the Choose database version selection.

Because the clone operation requires some time to complete, only perform read operations on the source database during the clone operation, or stop the database before you start the clone operation, to assure that the source database and the target database are consistent.
See Clone an Autonomous Data Warehouse Instance for the steps for this type of clone.

**Note:**

The clone has its own wallet and you cannot use the wallet of the source database to connect to the clone. All users must obtain a new wallet and use the new wallet to access the upgraded database. See Download Client Credentials (Wallets) for more information.

### Manually Upgrade by Cloning from a Backup

If your Autonomous Database instance is using Oracle Database 18c you can upgrade to Oracle Database 19c by creating a clone from a backup.

To upgrade your Autonomous Database by creating a clone from a backup, do the following:

1. Verify the Oracle Database version for your Autonomous Database instance is Oracle Database 18c.
   
   On the Oracle Cloud Infrastructure console, the Autonomous Database details page includes the **Database Version** field showing your database version.

2. During the cloning operation, in the Clone Type area select **Full Clone**, and in the Clone source area, select **Clone from a backup**.

3. During the cloning operation, in the cloning step where you configure the database, select the newer Oracle Database version in the **Choose database version** selection.

See Clone Autonomous Data Warehouse from a Backup for the steps for this type of clone.

**Note:**

The clone has its own wallet and you cannot use the wallet of the source database to connect to the clone. All users must obtain a new wallet and use the new wallet to access the upgraded database. See Download Client Credentials (Wallets) for more information.

### Automatically Upgrade to Oracle Database 19c

If your paid Autonomous Database instance is on Oracle Database 18c, after notifying you of the upgrade date, Oracle will automatically upgrade the Autonomous Database instance to Oracle Database 19c (you can manually upgrade to Oracle Database 19c before the automatic upgrade date).

Autonomous Database automatic upgrade involves the following:

- Oracle will send notifications for an upcoming automatic upgrade starting at least two months prior to the automatic upgrade date.
• The automatic upgrade will be performed on the announced date and in your database's maintenance window. You can see the maintenance window for your database on the Autonomous Database Details page.

• The automatic upgrade is a minimal downtime operation and requires up to ten (10) minutes of downtime during the maintenance window.

• After the automatic upgrade is completed, the database version on the Autonomous Database details page will show Oracle Database 19c. At this point, you can connect to the database using your existing wallets and credentials and continue to use the database.
Wallet Security, Data Safe, and Oracle Database Vault

This section describes options for rotating wallets, using Oracle Data Safe service, configuring Oracle Database Vault, and describes how you can change your license or account type.

Topics

• Rotate Wallets for Autonomous Database
• Update License Type on Autonomous Database
• Update Always Free Instance to Paid with Autonomous Database
• Safeguard Your Data with Data Safe on Autonomous Database
• Use Oracle Database Vault with Autonomous Database

Rotate Wallets for Autonomous Database

Wallet rotation lets you invalidate existing client certification keys for a database instance or for all Autonomous Database instances in a region.

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.

There are two options for 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.

• 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.
To rotate the client certification key for a given database or for all Autonomous Databases in a region:

1. Navigate to the Autonomous Database details page.
2. Click **DB 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.

If you want to terminate all connections immediately after the wallet rotation completes, Oracle recommends that you stop and then start your instance. This provides the highest level of security for your database. See Stop Autonomous Data Warehouse for more information.

Note:
Oracle recommends you provide a database-specific instance wallet, with Wallet Type set to Instance Wallet when you use Download 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.
You can also use the Autonomous Database API to rotate wallets using `UpdateAutonomousDatabaseRegionalWallet` and `UpdateAutonomousDatabaseWallet`. See Autonomous Database Wallet Reference for more information.

### Update License Type on Autonomous Database

Describes how to update your licensing with Autonomous Database.

- Sign in to your Oracle Cloud Account at [cloud.oracle.com](http://cloud.oracle.com).
- From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
- On the Autonomous Databases page select an Autonomous Data Warehouse instance from the links under the **Display Name** column.

1. On the **Details** page, from the **More Actions** drop-down list, select **Update License Type**.
2. On the **Update License Type** page select the license type:
   - My organization already owns Oracle database software licenses
     Bring my existing database software licenses to the database cloud service
   - Subscribe to new database software licenses and the database cloud service
3. Click **Update**.

### 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.

Next, upgrade your Always Free Autonomous Database instance as follows:

- Sign in to your Oracle Cloud Account at [cloud.oracle.com](http://cloud.oracle.com).
- From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.
- Select the Always Free instance by clicking the Name link.

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.
Safeguard Your Data with Data Safe on Autonomous Database

Oracle Data Safe provides features that help you protect sensitive and regulated data in your Autonomous Data Warehouse database.

See Oracle Data Safe Overview for an overview of Oracle Data Safe.

See Enable Oracle Data Safe in a Region for information on enabling Oracle Data Safe.

Use Oracle Database Vault with Autonomous Database

Oracle Database Vault implements powerful security controls for your Autonomous Data Warehouse 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.

Topics

• Oracle Database Vault Users and Roles on Autonomous Database
• Enable Oracle Database Vault on Autonomous Database
• Disable Oracle Database Vault on Autonomous Database
• Disable User Management with Oracle Database Vault on Autonomous Database
• Enable User Management with Oracle Database Vault on Autonomous Database

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:

   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 Data Warehouse instance.

   See Restart Autonomous Data Warehouse 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.
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 Data Warehouse instance.

   See Restart Autonomous Data Warehouse 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:

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DV_CONFIGURE_STATUS</td>
<td>TRUE</td>
</tr>
<tr>
<td>DV_ENABLE_STATUS</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

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 Application Express(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 Application Express (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:

   ```sql
   EXEC DBMS_CLOUD_MACADM.ENABLE_USERMGMT_DATABASE_VAULT('APEX');
   ```

See `ENABLE_USERMGMT_DATABASE_VAULT Procedure` for more information.
Configuring Network Access with Access Control Rules (ACLs) and Private Endpoints

This chapter describes how to configure network access with access control rules or using a private endpoint.

Topics

• Configure an Access Control List with Autonomous Database
• Configure Private Endpoints with Autonomous Database
• Private Endpoint Configuration Examples on Autonomous Database

Configure an Access Control List with Autonomous Database

You can control and restrict access to your Autonomous Database by setting network access control lists (ACLs).

To add an ACL when you provision your Autonomous Database, under Choose network access select Configure access control rules and then enter rules as described in Step 2, Step 3, and Step 4.

See Provision Autonomous Data Warehouse for information on provisioning your Autonomous Database.

To add or change ACLs for an Autonomous Database instance, do the following:

• Sign in to your Oracle Cloud Account at cloud.oracle.com.
• From the Oracle Cloud Infrastructure left navigation list click Autonomous Data Warehouse.

• On the Autonomous Databases page select an Autonomous Data Warehouse instance from the links under the Display Name column.

1. On the Details page, from the More Actions drop-down list, select Access Control List.

2. On the Edit Access Control List page select from the choices:
   • **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.
   
   • **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 whitelist 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 whitelist specific clients in the VCN.
3. Click + Another Entry 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 Save Changes.

   If the Lifecycle State is Available when you click Save Changes 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.

Access Control List Notes:

- 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 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.

- Oracle Application Express (APEX), RESTful services, and SQL Developer Web 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.

- The Autonomous Data Warehouse Service console is not subject to ACLs.
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.

Configure Private Endpoints with Autonomous Database

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. You can configure a private endpoint when you provision or clone your Autonomous Database; this allows you to keep all traffic to and from your Autonomous Database off the public internet.

To configure a private endpoint do the following before you provision your Autonomous Database:

- 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.
- Specify at least one Network Security Group (NSG) within your VCN. The NSG specifies rules for connections to your Autonomous Database. See Network Security Groups for more information.

During provisioning or cloning, to specify a private endpoint for your Autonomous Database do the following:

1. Under Choose network access, select Virtual cloud network.
   
   This expands the Virtual cloud network private access configuration area.
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) 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:

\texttt{hostname\_prefix.adwc.region.oraclecloud.com}

If you do not specify a hostname prefix, a system generated hostname prefix is supplied.

5. Select one or more \textbf{Network security groups (NSGs)}. To allow connections to the Autonomous Database instance, you need to define security rules in an NSG; this creates a virtual firewall for your Autonomous Database.

For the NSG you select for the private endpoint define a security rule as follows:

- 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 \textit{Private Endpoint Configuration Examples on Autonomous Database} for examples.

See \textit{Network Security Groups} for more information.

6. Complete the remaining provisioning or cloning steps, as specified in \textit{Provision Autonomous Data Warehouse}, \textit{Clone an Autonomous Data Warehouse Instance}, or \textit{Clone Autonomous Data Warehouse from a Backup}.

7. After the provisioning or cloning completes, you can view the network configuration on the Autonomous Database Details page under the \textbf{Network} section.

The \textbf{Network} section shows the following information for a private endpoint:

- **Access Type**: Specified the access type for the Autonomous Database configuration. Private endpoint configurations show: \textit{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.

Private endpoint restrictions and notes:

- If you want to use a private endpoint this must be configured when you provision the database. After you create an Autonomous Database you cannot change the selection you make, either \textit{Allow secure access from everywhere} or \textit{Virtual cloud network} for the Autonomous Data Warehouse database instance. However, you can clone a new database and change the selection for the new cloned database.

- You can specify up to five NSGs to control access to your Autonomous Database.

- You can change the Network Security Group (NSG) that you specify when you provision your Autonomous Database.

To change the NSG for a private endpoint, do the following:
1. On the Autonomous Databases page select an Autonomous Data Warehouse instance 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**.
   - Data Safe is not supported with private endpoints. See [Safeguard Your Data with Data Safe on Autonomous Database](#) for information on Using Data Safe without private endpoints.
   - 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 [Connect to On-premise Data Sources](#) for more information.
   - You cannot change the VCN or the Subnet that you specify when you provision your Autonomous Database (except if you clone a new database).
   - Oracle Application Express, Oracle SQL Developer Web, Oracle Machine Learning Notebooks, and Oracle REST Data Services are not supported in databases with private endpoints.

**Private Endpoint Configuration Examples on Autonomous Database**

Shows several Private Endpoint (VCN) configuration samples for Autonomous Database.

This section includes the following:

- **Sample 1: Connecting from Inside Oracle Cloud Infrastructure VCN**
- **Sample 2: Connecting from Your Data Center to Autonomous Database**

**Sample 1: Connecting from Inside Oracle Cloud Infrastructure VCN**

This example 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:

- 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.

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.


Sample 2: Connecting from Your Data Center to Autonomous Database

This example 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 host's /etc/hosts file. For example:

```
/etc/hosts entry -> 10.0.2.7 example.adb.ca-toronto-1oraclecloud.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 set up a hybrid DNS in Oracle Cloud Infrastructure for DNS name resolution.

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:

- 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.

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.
Maintenance Windows, Work Requests and Managing Events and Notifications

Oracle Cloud Infrastructure Events enable you to create automation based on state changes for resources. Oracle Cloud Infrastructure work requests allow you to monitor long-running operations such as cloning or backing up an Autonomous Data Warehouse database. The Oracle Cloud Infrastructure Console provides information on Autonomous Database maintenance windows.

Topics

• About Events and Notifications
• About Work Requests
• About Autonomous Database Maintenance Windows

About Events and Notifications

An event could be a resource lifecycle state change or a system event impacting a resource. For example, an event can be emitted when a backup or restore operation begins or ends. You can use events to manage actions, including notifications and other actions.

See Overview of Events for more information.

About Work Requests

Autonomous Databases support Oracle Cloud Infrastructure Work Requests. 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
• Creating or deleting manual backups.
• 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.
About Autonomous Database Maintenance Windows

Autonomous Database uses predefined maintenance windows to automatically patch your database.

All Autonomous Database instances are automatically assigned to a maintenance window and different instances can have different maintenance windows. On the Autonomous Database details page, the **Next Maintenance** field shows the date and time for the upcoming maintenance window; this field is updated automatically when the next maintenance window is scheduled.

**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.
Chapter 20

About Autonomous Database Maintenance Windows

Sales Autonomous Data Warehouse

General Information
- Database name: OE201
- Workload Type: Data Warehouse
- Compartiment: (root)
- OCID: Show
- Created: Wed, Dec 12, 2018, 18:08:36 UTC
- OCPU Count: 1
- Storage (TB): 1
- License Type: Bring Your Own License (BYOL)
- Database Version: 18c
- Auto Scaling: Disabled
- Lifecycle State: Available
- Instance Type: Paid

Infrastructure
- Dedicated Infrastructure: No

Backup
- Last Automatic Backup: No active backups exist for this database.

Network
- Access Type: Allow secure access from anywhere
- Access Control List: Disabled

Maintenance
- Next: Sun, Dec 1, 2019, 06:00:00 UTC
- Maintenance: 06:00:00 UTC
Using a Standby Database with Autonomous Database

Autonomous Database provides Autonomous Data Guard where you can choose to enable a standby (peer) database to provide data protection and disaster recovery for your Autonomous Database instance.

Topics

• About Autonomous Data Guard
• Enable Autonomous Data Guard
• Disable Autonomous Data Guard
• Switchover Autonomous Data Guard
• Manual Failover with Autonomous Data Guard

About Autonomous Data Guard

When you enable Autonomous Data Guard, the system creates a standby database that stays "connected" to the primary database and automatically refreshes its data from the primary database.

With Autonomous Data Guard enabled Autonomous Database provides one 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 the standby database. You perform all operations, such as scaling up the OCPU Count and enabling Auto Scaling on the primary database and Autonomous Data Guard then performs the same actions on the standby database. Likewise, you only perform actions such as stopping or restarting the database on the primary database.

Autonomous Data Guard Features

When you enable Autonomous Data Guard on your Autonomous Database instance, a standby database is provisioned and is updated with data to reflect the primary. Autonomous Data Guard monitors the primary database and if the Autonomous Database instance goes down then the standby instance assumes the role of the primary instance.
The standby database is created in the same region as the primary database. For better resilience, the standby database is provisioned as follows:

- In regions with more than one availability domain, the standby database is provisioned in a different availability domain than the primary database.
- In regions with a single availability domain, the standby database is provisioned on a different physical machine than the primary database.

See Regions and Availability Domains for more information.

All Autonomous Database features from the primary database are available when the standby instance becomes the primary after the system fails over or after you perform a switchover operation, including the following:

- **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.

All primary database options are available when the standby instance becomes the primary after a failover operation or after you perform a switchover, including the following:

- The Compartment, OCPU Count, Storage, Display Name, Database Name, Auto Scaling, Tags, and Licensing options. Any updates made to these primary database options apply in the database after a failover operation. For example if the OCPU Count is scaled up and Auto Scaling is enabled for the primary database, the same actions occur on the standby database.
- Any APIs or scripts for managing the Autonomous Database continue to work without any changes after a failover operation or after you perform a switchover.
- 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.
- 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 Database Failover with Autonomous Data Guard**

After you enable Autonomous Data Guard, the system monitors the primary instance and automatically fails over to the standby database in certain scenarios. If automatic failover is not possible, you have the option to perform a manual failover.

Autonomous Database fails over to the standby database as follows:

- When the primary database becomes unavailable and users are not able to connect to it, Autonomous Data Guard attempts to automatically fail over to the standby database based on the Recovery Time Objective (RTO) and Recovery Point Objective (RPO) targets. This automatic failover operation is guaranteed to provide zero data loss.
• If Autonomous Data Guard cannot automatically fail over to the standby database because of the risk of data loss, then you can perform a manual failover operation. In this case Oracle Cloud Infrastructure console shows a banner indicating that the automatic failover did not succeed, and provides a link for you to initiate a manual failover.

• If the primary database has failed or is unreachable and the conditions for Autonomous Data Guard automatic failover have not been met, you can perform a manual failover.

• After failover completes, Autonomous Database creates a new standby database for you. Autonomous Data Guard is not enabled when the system is provisioning the new standby database and the Peer State field shows Provisioning. After Autonomous Data Guard completes the provisioning step, then you have a new standby database and Autonomous Data Guard is enabled.

Autonomous Data Guard Operations

Autonomous Database provides the following operations with Autonomous Data Guard:

• Enable: If Autonomous Data Guard is disabled, you can enable Autonomous Data Guard. See Enable Autonomous Data Guard for details.

• Disable: If Autonomous Data Guard is enabled, you can disable Autonomous Data Guard. Disabling Autonomous Data Guard terminates the standby database. See Disable Autonomous Data Guard for details.

• Manual Switchover: When Autonomous Data Guard is enabled, switchover changes the roles of the primary and standby, your standby database becomes the primary, your primary database becomes the standby. See Switchover Autonomous Data Guard for details.

• Manual Failover: You can attempt a manual failover to make the standby database be the primary database. You can only perform a manual failover when the primary database is not available. See Manual Failover with Autonomous Data Guard for details.

• Terminate: If you want to terminate a primary instance, select More Actions and Terminate. Terminating the primary instance also terminates the standby database.

Autonomous Database Standby Database States

Autonomous Database provides information about Autonomous Data Guard on the Autonomous Database Details page. This page shows Autonomous Data Guard information, including the status, enabled or disabled, and the state of the standby database.

The Status field shows the Autonomous Data Guard status information, as follows:

• Enabled indicates Autonomous Data Guard is enabled.

• Disabled indicates Autonomous Data Guard is not enabled.

The Peer State field shows the state of the standby database, as follows:

• Provisioning
This state shows when you enable Autonomous Data Guard. This status is shown to indicate the standby database is provisioning (until the standby database is available).

This state shows after a failover when the standby database is being recreated.

If a restore from backup operation is performed on the primary database, then the standby is recreated and the Peer State field shows Provisioning.

- **Available** Indicates that the standby is available and ready for either a switchover or a failover operation.

### Autonomous Data Guard Notes

Note the following for using Autonomous Database with Autonomous Data Guard:

- Autonomous Data Guard is not available with Always Free Autonomous Databases.

- Autonomous Database does not provide access to the standby database:
  - You perform all operations, such as scaling up the OCPU Count and enabling Auto Scaling on the primary database and Autonomous Database performs the same actions on the standby database. Likewise, you only perform actions such as stopping or restarting the database on the primary database.
  - The standby database is not available for use as a read-only database.

- The Number of OCPUs allocated graph and the CPU utilization graph on the Autonomous Database Service Console display the OCPUs allocated and the CPUs utilization for the primary database. These graphs do not include information about the standby database.

  The CPU Utilization metrics on the Oracle Cloud Infrastructure console metrics page displays 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 standby database.

- You cannot enable Autonomous Data Guard on Autonomous Database if your database version is Oracle Database 18c. You must first upgrade to Oracle Database 19c or higher to use Autonomous Data Guard. See Upgrade an Autonomous Database Instance to Oracle Database 19c for more information.

### Enable Autonomous Data Guard

You can enable Autonomous Data Guard when the Autonomous Database is available (Lifecycle State shows Available).

**Note:**

To enable Autonomous Data Guard you must have adequate available resources. Thus, by enabling Autonomous Data Guard you must not exceed your Tenancy or compartment limits for CPU and Storage.

To enable Autonomous Data Guard do the following:
• Sign in to your Oracle Cloud Account at cloud.oracle.com.

• From the Oracle Cloud Infrastructure left navigation list click either Autonomous Data Warehouse or Autonomous Transaction Processing, depending on the workload type of your instance.

• On the Autonomous Databases page select your instance from the links under the Display Name column.

1. On the Details page, under Autonomous Data Guard, click **Enable** in the **Status** field.

   ![Enable Autonomous Data Guard dialog](image)

   Enabling Autonomous Data Guard creates a peer database. Your tenancy will be billed for OCPU and storage resources utilized by the peer database. Do you you want to enable Autonomous Data Guard?

   - Enable Autonomous Data Guard
   - Cancel

2. In the Enable Autonomous Data Guard dialog, click **Enable Autonomous Data Guard**.

   On the Details page, under Autonomous Data Guard the Peer State field shows Provisioning. After some time, the Status field shows Enabled and provisioning continues.

   When provisioning completes, the Peer State field shows Available and shows the Switchover link.

   ![Autonomous Data Guard status](image)

   **Status:** Enabled  **Disable**

   **Peer State:** Available  **Switchover**

Notes for enabling Autonomous Data Guard:

• To enable Autonomous Data Guard the database version must be Oracle Database 19c or higher.

• When you enable Autonomous Data Guard the database is available and you can connect to the database and you can use the database, with the following restrictions.

   While you enable Autonomous Data Guard, when the Peer State field shows Provisioning, the following actions are disabled for the database:
   
   - Move Resource. See [Move an Autonomous Data Warehouse Database to a Different Compartment](#) for information on moving an instance.
   
   - Stop. See [Stop Autonomous Data Warehouse](#) for information on stopping an instance.
– Restart. See Restart Autonomous Data Warehouse for information on
restarting an instance.
– Restore. See Restore and Recover your Autonomous Data Warehouse
Database for information on restoring.

Disable Autonomous Data Guard

Describes the steps to disable Autonomous Data Guard.

• Sign in to your Oracle Cloud Account at cloud.oracle.com.
• From the Oracle Cloud Infrastructure left navigation list click either Autonomous
Data Warehouse or Autonomous Transaction Processing, depending on the
workload type of your instance.
• On the Autonomous Databases page select your instance from the links under the
Display Name column.

1. On the Details page, under Autonomous Data Guard, in the Status field, click
Disable.

2. In the Disable Autonomous Data Guard dialog, enter the Autonomous Database
name to confirm that you want to disable Autonomous Data Guard for the
instance.

3. In the Disable Autonomous Data Guard dialog, click Disable Autonomous Data
Guard.

This disables Autonomous Data Guard for the instance and terminates the standby
database.
Switchover Autonomous Data Guard

When Autonomous Data Guard is enabled, when you perform a switchover operation the primary database becomes the standby database, and the standby database becomes the primary database.

**Note:**

You can only use switchover when the Autonomous Data Guard primary instance is in Lifecycle State Available or Stopped and the standby instance is in the Available state.

- Sign in to your Oracle Cloud Account at cloud.oracle.com.
- From the Oracle Cloud Infrastructure left navigation list click either Autonomous Data Warehouse or Autonomous Transaction Processing, depending on the workload type of your instance.
- On the Autonomous Databases page select your instance from the links under the **Display Name** column.

1. On the Details page, under Autonomous Data Guard, in the **Peer State** field, click **Switchover**.

   ![Confirm Manual Switchover to Standby](image)

   **Note:**
   When you perform a switchover operation, the dialog shows any possible data loss. The \( x \) minutes of data loss is a value, for example 3 (minutes), that depends on the current lag time of the standby database.

2. In the Confirm Manual Switchover to Standby dialog, enter the Autonomous Database name to confirm that you want to switchover.

3. In the Confirm Manual Switchover to Standby dialog, click **Confirm Manual Switchover to Standby**.
Note for Autonomous Data Guard switchover operations:

- During the switchover most of the actions on the Oracle Cloud Infrastructure console are not available and the Autonomous Database Information tab shows the Lifecycle State Updating.

Manual Failover with Autonomous Data Guard

Describes the steps to perform a manual failover.

When Autonomous Data Guard cannot automatically fail over to the standby database you can perform a manual failover in certain conditions to make the standby database the primary database. You can also perform a manual failover operation in cases where Autonomous Data Guard has not attempted to automatically fail over to the standby database and the primary database has failed or is unreachable.

You can perform a manual failover only when the primary database is not available, that is the primary database is not showing the Lifecycle State Available or Stopped and the standby (peer) is in the Available state.

- Sign in to your Oracle Cloud Account at cloud.oracle.com.
- From the Oracle Cloud Infrastructure left navigation list click either Autonomous Data Warehouse or Autonomous Transaction Processing, depending on the workload type of your instance.
- On the Autonomous Databases page select your instance from the links under the Display Name column.
  1. On the Details page, under Autonomous Data Guard, in the Peer State field, click Failover.
  2. In the Confirm Manual Failover dialog, enter the Autonomous Database name to confirm that you want to failover.

Note:

Autonomous Data Guard creates a new standby database after the failover. Autonomous Data Guard is not enabled when the system is provisioning the new standby database and the Peer State field shows Provisioning. After Autonomous Data Guard completes the provisioning step, then you have a new standby database and Autonomous Data Guard is enabled.
Part III
Appendixes

Part III contains the Appendixes.

Topics

• Autonomous Database Supplied Package Reference
• Autonomous Data Warehouse for Experienced Oracle Database Users
• Migrating Amazon Redshift to Autonomous Data Warehouse
• Sample Star Schema Benchmark (SSB) Queries and Analytic Views
• SODA Collection Metadata on Autonomous Database
• Obtain Tenancy Details to File a Service Request
Autonomous Database Supplied Package Reference

This section provides information about the DBMS_CLOUD, DBMS_CLOUD_ADMIN, DBMS_CLOUD_MACADM, and DBMS_MAX_STRING_SIZE packages you use with Autonomous Database. This also covers the DBMS_CLOUD REST APIs.

Topics

- Summary of DBMS_CLOUD Subprograms
- DBMS_CLOUD Package File URI Formats
- DBMS_CLOUD Package Format Options
- DBMS_CLOUD Package Format Options for Parquet and Avro
- DBMS_CLOUD Package Parquet to Oracle Data Type Mapping
- DBMS_CLOUD Package Avro to Oracle Data Type Mapping
- DBMS_CLOUD Package Parquet and AVRO to Oracle Column Name Mapping
- Summary of DBMS_CLOUD_ADMIN Subprograms
- Summary of DBMS_MAX_STRING_SIZE Subprograms
- Summary of DBMS_CLOUD_MACADM Subprograms
- DBMS_CLOUD REST APIs

Summary of DBMS_CLOUD Subprograms

This section covers the DBMS_CLOUD subprograms 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 adwc_user:

```
GRANT EXECUTE ON DBMS_CLOUD TO adwc_user;
```

Topics

- COPY_COLLECTION Procedure
COPY_COLLECTION Procedure

This procedure loads data into existing SODA collection from Cloud Object Storage. 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>collection_name</td>
<td>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.</td>
</tr>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>file_uri_list</td>
<td>Comma-delimited list of source file URIs. You can use wildcards in the file names in your URIs. The character &quot;*&quot; can be used as the wildcard for multiple characters, the character &quot;?&quot; can be used as the wildcard for a single character. The format of the URIs depends on the Cloud Object Storage service. For details see DBMS_CLOUD Package File Cloud Object Storage URI Formats.</td>
</tr>
<tr>
<td>format</td>
<td>The options describing the format of the source files. These options are specified as a JSON string. Supported formats for JSON data are: unpackarrays, recorddelimiter, ignoreblanks, characterset, rejectlimit, compression. 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 Package Format Options.</td>
</tr>
<tr>
<td>operation_id</td>
<td>Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.</td>
</tr>
</tbody>
</table>

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/adwc4pm/b/json/o/myCollection.json'
    );
END;
/

COPY_DATA Procedure

This procedure loads data into existing Autonomous Database tables from files in the Cloud. The overloaded form enables you to use the operation_id parameter.
Syntax

DBMS_CLOUD.COPY_DATA ( 
  table_name        IN VARCHAR2, 
  credential_name   IN VARCHAR2, 
  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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>The name of the target table on the database. The target table needs to be created before you run COPY_DATA.</td>
</tr>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>file_uri_list</td>
<td>Comma-delimited list of source file URIs. You can use wildcards in the file names in your URIs. The character &quot;*&quot; can be used as the wildcard for multiple characters, the character &quot;?&quot; can be used as the wildcard for a single character. The format of the URIs depend on the Cloud Object Storage service you are using, for details see DBMS_CLOUD Package File URI Formats.</td>
</tr>
<tr>
<td>schema_name</td>
<td>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.</td>
</tr>
<tr>
<td>field_list</td>
<td>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. For an example using field_list, see CREATE_EXTERNAL_TABLE Procedure.</td>
</tr>
<tr>
<td>format</td>
<td>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 Parquet and Avro file format options, see DBMS_CLOUD Package Format Options for Parquet and Avro.</td>
</tr>
<tr>
<td>operation_id</td>
<td>Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.</td>
</tr>
</tbody>
</table>
COPY_DATA Procedure for Parquet or Avro Files

This procedure with the format parameter type set to the value parquet or avro loads data into existing Autonomous Database tables from Parquet or Avro files in the Cloud. Similar to text files, the data is copied from the source Parquet or Avro file into the preexisting internal table.

Syntax

```sql
DBMS_CLOUD.COPY_DATA (
    table_name        IN VARCHAR2,
    credential_name   IN VARCHAR2,
    file_uri_list     IN CLOB,
    schema_name       IN VARCHAR2 DEFAULT,
    field_list        IN CLOB DEFAULT,
    format            IN CLOB DEFAULT);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>The name of the target table on the database. The target table needs to be created before you run COPY_DATA.</td>
</tr>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>file_uri_list</td>
<td>Comma-delimited list of source file URIs. You can use wildcards in the file names in your URIs. The character &quot;*&quot; can be used as the wildcard for multiple characters, the character &quot;?&quot; can be used as the wildcard for a single character. The format of the URIs depend on the Cloud Object Storage service you are using, for details see DBMS_CLOUD Package File URI Formats.</td>
</tr>
<tr>
<td>schema_name</td>
<td>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.</td>
</tr>
<tr>
<td>field_list</td>
<td>Ignored for Parquet and Avro 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 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.</td>
</tr>
<tr>
<td>format</td>
<td>The options describing the format of the source files. For Parquet and Avro files, only two options are supported: see DBMS_CLOUD Package Format Options for Parquet and Avro.</td>
</tr>
</tbody>
</table>

Usage Notes

As with other data files, 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 Parquet or Avro files, when the `format` parameter type is set to the value `parquet` or `avro`, 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.

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. See CREATE_CREDENTIAL Procedure (OCI Signing Key Credentials) for more information.

**Syntax**

```
DBMS_CLOUD.CREATE_CREDENTIAL (  
    credential_name   IN VARCHAR2,  
    username          IN VARCHAR2,  
    password          IN VARCHAR2 DEFAULT NULL);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>credential_name</code></td>
<td>The name of the credential to be stored.</td>
</tr>
<tr>
<td><code>username</code></td>
<td>The <code>username</code> and <code>password</code> arguments together specify your cloud service credentials. See below for what to specify for the <code>username</code> and <code>password</code> for different cloud services.</td>
</tr>
<tr>
<td><code>password</code></td>
<td>The <code>username</code> and <code>password</code> arguments together specify your cloud service credentials.</td>
</tr>
</tbody>
</table>

**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. See CREATE_CREDENTIAL Procedure (OCI Signing Key Credentials) for more information.

Oracle Cloud Infrastructure Credentials (Auth Tokens)

For Oracle Cloud Infrastructure the username is your Oracle Cloud Infrastructure username. The password is your Oracle Cloud Infrastructure auth token. See Working with Auth Tokens.

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.

CREATE_CREDENTIAL Procedure (OCI Signing Key Credentials)

This procedure stores Oracle Cloud Infrastructure (OCI) Cloud Service credentials in the Autonomous Database.

Use stored credentials for data loading or for querying external data residing in the Cloud, where 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 Key based authentication. To create credentials for other cloud services, with the username and password parameters, see CREATE_CREDENTIAL Procedure.

Syntax

```sql
DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name IN VARCHAR2,
    user_ocid       IN VARCHAR2,
    tenancy_ocid    IN VARCHAR2,
    private_key     IN VARCHAR2,
    fingerprint     IN VARCHAR2);
```
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>The name of the credential to be stored.</td>
</tr>
<tr>
<td>user_ocid</td>
<td>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.</td>
</tr>
<tr>
<td>tenancy_ocid</td>
<td>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.</td>
</tr>
<tr>
<td>private_key</td>
<td>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.</td>
</tr>
<tr>
<td>fingerprint</td>
<td>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.</td>
</tr>
</tbody>
</table>

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. Also see CREATE_CREDENTIAL Procedure for more information.
- 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.

CREATE_EXTERNAL_TABLE Procedure

This procedure creates an external table on files in the Cloud. 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,  
    file_uri_list    IN CLOB,  
    column_list      IN CLOB,  
    field_list       IN CLOB DEFAULT,  
    format           IN CLOB DEFAULT);  
```
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>The name of the external table.</td>
</tr>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>file_uri_list</td>
<td>Comma-delimited list of source file URIs. You can use wildcards in the file names in your URIs. The character &quot;*&quot; can be used as the wildcard for multiple characters, the character &quot;?&quot; can be used as the wildcard for a single character. The format of the URIs depend on the Cloud Object Storage service you are using, for details see DBMS_CLOUD Package File URI Formats.</td>
</tr>
<tr>
<td>column_list</td>
<td>Comma-delimited list of column names and data types for the external table.</td>
</tr>
<tr>
<td>field_list</td>
<td>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.</td>
</tr>
<tr>
<td>format</td>
<td>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 Parquet or Avro format files, see CREATE_EXTERNAL_TABLE Procedure for Parquet or Avro Files.</td>
</tr>
</tbody>
</table>

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, Microsoft Azure, and AWS S3. The credential is a table level property; therefore, the external files must be on the same object store. See DBMS_CLOUD Package File URI Formats for more information.

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',
        column_list => 'REPORT_DATE DATE ''mm/dd/yy'',
        field_list => '
            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 Parquet or Avro Files

This procedure with the format parameter type set to the value parquet or avro creates an external table with either Parquet or Avro format files in the Cloud. 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,
file_uri_list IN CLOB,
column_list IN CLOB,
field_list IN CLOB DEFAULT,
format IN CLOB DEFAULT);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>The name of the external table.</td>
</tr>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>file_uri_list</td>
<td>Comma-delimited list of source file URIs. You can use wildcards in the file names in your URIs. The character &quot;*&quot; can be used as the wildcard for multiple characters, the character &quot;?&quot; can be used as the wildcard for a single character. The format of the URIs depend on the Cloud Object Storage service you are using, for details see DBMS_CLOUD Package File URI Formats.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>column_list</td>
<td>(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 a Parquet or Avro source, the column names must match those columns found in the file. Oracle data types must map appropriately to the Parquet or Avro data types. 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.</td>
</tr>
<tr>
<td>field_list</td>
<td>Ignored for Parquet or Avro 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 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.</td>
</tr>
<tr>
<td>format</td>
<td>For Parquet and Avro, there are only two supported parameters. See DBMS_CLOUD Package Format Options for Parquet and Avro for details.</td>
</tr>
</tbody>
</table>

**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 and Parquet Column Name Mapping to Oracle Column Names**

See DBMS_CLOUD Package Parquet and AVRO to Oracle Column Name Mapping for information on column name mapping and column name conversion usage in Oracle SQL.

**CREATEEXTERNALPART_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.
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);  

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>The name of the external table.</td>
</tr>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>partitioning_clause</td>
<td>Specifies the complete partitioning clause, including the location information for individual partitions.</td>
</tr>
<tr>
<td>column_list</td>
<td>Comma-delimited list of column names and data types for the external table.</td>
</tr>
<tr>
<td>field_list</td>
<td>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.</td>
</tr>
<tr>
<td>format</td>
<td>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.</td>
</tr>
</tbody>
</table>

Usage Notes

- With Parquet or Avro data format, using DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE, the schema format option is not available and the column_list parameter must be specified. The schema format option is available with DBMS_CLOUD.CREATE_EXTERNAL_TABLE.
- 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, Microsoft Azure, and AWS S3. The credential is a table level property; therefore, the external files must be on the same object store. See DBMS_CLOUD Package File URI Formats for more information.

Example

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)
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. Hybrid partitioned tables are only supported with Oracle Database 19c onwards.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>The name of the external table.</td>
</tr>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>partitioning_clause</td>
<td>Specifies the complete partitioning clause, including the location information for individual partitions.</td>
</tr>
<tr>
<td>column_list</td>
<td>Comma-delimited list of column names and data types for the external table.</td>
</tr>
<tr>
<td>field_list</td>
<td>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.</td>
</tr>
<tr>
<td>format</td>
<td>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.</td>
</tr>
</tbody>
</table>
Usage Note

- 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, Microsoft Azure, and AWS S3. The credential is a table level property; therefore, the external files must be on the same object store. See DBMS_CLOUD Package File URI Formats for more information.

Example

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;
/

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>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.</td>
</tr>
</tbody>
</table>
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**

```sql
DBMS_CLOUD.DELETE_FILE (
    directory_name     IN VARCHAR2,
    file_name          IN VARCHAR2);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>directory_name</td>
<td>The name of the directory on the Autonomous Database instance.</td>
</tr>
<tr>
<td>file_name</td>
<td>The name of the file to be removed.</td>
</tr>
</tbody>
</table>

**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 **adwc_user**:

```
GRANT WRITE ON DIRECTORY data_pump_dir TO adwc_user;
```

**DELETE_OBJECT Procedure**

This procedure deletes the specified object on object store.

**Syntax**

```sql
DBMS_CLOUD.DELETE_OBJECT (
    credential_name      IN VARCHAR2,
    object_uri           IN VARCHAR2);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>object_uri</td>
<td>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 Package File URI Formats.</td>
</tr>
</tbody>
</table>

**DROP_CREDENTIAL Procedure**

This procedure removes an existing credential from Autonomous Database.

**Syntax**

```
DBMS_CLOUD.DROP_CREDENTIAL (credential_name IN VARCHAR2);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>The name of the credential to be removed.</td>
</tr>
</tbody>
</table>

**EXPORT_DATA Procedure**

This procedure exports data from Autonomous Database to Oracle Data pump files in the Cloud based on the result of the specified SQL query. Using this procedure Autonomous Database uses the ORACLE_DATAPUMP access driver to write data to a dump file(s) on the Cloud Object store. The overloaded form enables you to use the `operation_id` parameter.

**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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>file_uri_list</td>
<td>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. The format of the URIs depend on the Cloud Object Storage service you are using, for details see DBMS_CLOUD Package File URI Formats.</td>
</tr>
</tbody>
</table>
| format        | Specify export format options. Supported options are:  
• TYPE: The TYPE format parameter is required and must have the value DATAPUMP.  
In addition, with the format parameter you can specify supported Oracle Data Pump access parameters. The supported Oracle Data Pump access parameters are:  
• 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. |
| query         | Use this parameter to specify a SELECT statement so that only the required data is exported. The query determines the contents of the dump file(s). For example:  
'SELECT warehouse_id, quantity FROM inventories'  
See Oracle Data Pump Export Data Filters and Unloading and Loading Data with the ORACLE_DATAPUMP Access Driver for more information. |
| 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

- Autonomous Database export using DBMS_CLOUD.EXPORT_DATA only supports Oracle Cloud Infrastructure Object Storage and Oracle Cloud Infrastructure Object Storage Classic object stores.

- The Oracle Cloud Infrastructure Object Store console shows multiple files for each dump file that you export, and the size of the actual dump files will be displayed as zero (0). For example:

```
exp01.dmp
exp01.dmp_aaaaaa
exp02.dmp
exp02.dmp_aaaaaa
```

Oracle Data Pump divides each dump file into smaller chunks for faster uploads. Downloading the zero byte dump file from the console or Oracle Cloud Infrastructure CLI does not give you the full dump file. To download the dump files...
from the Object Store, use a tool that supports Swift such as `curl`, and provide your user login and Swift auth token. For example, `curl` with `GET`:

```
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 instance on Shared Infrastructure, 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.
Example

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.

GET_OBJECT Procedure

This procedure reads an object from Cloud Object Storage and copies it to Autonomous Database. The maximum file size allowed in this procedure is 5 gigabytes (GB).

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);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>object_uri</td>
<td>Object or file URI. The format of the URI depends on the Cloud Object Storage service you are using, for details see DBMS_CLOUD Package File URI Formats.</td>
</tr>
<tr>
<td>directory_name</td>
<td>The name of the directory on the database.</td>
</tr>
<tr>
<td>file_name</td>
<td>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.</td>
</tr>
<tr>
<td>startoffset</td>
<td>The offset, in bytes, from where the procedure starts reading.</td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>endoffset</td>
<td>The offset, in bytes, until where the procedure stops reading.</td>
</tr>
<tr>
<td>compression</td>
<td>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).</td>
</tr>
</tbody>
</table>

#### 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 adwc_user:

```
GRANT WRITE ON DIRECTORY data_pump_dir TO adwc_user;
```

#### Example

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/cwallet.sso',  
    directory_name => 'DATA_PUMP_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.

### 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>directory_name</td>
<td>The name of the directory on the database.</td>
</tr>
</tbody>
</table>
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 adwc_user:

  GRANT READ ON DIRECTORY data_pump_dir TO adwc_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');
```

<table>
<thead>
<tr>
<th>OBJECT_NAME</th>
<th>BYTES</th>
<th>CHECKSUM</th>
<th>CREATED</th>
<th>LAST_MODIFIED</th>
</tr>
</thead>
<tbody>
<tr>
<td>cwallet.sso</td>
<td>2965</td>
<td></td>
<td>2018-12-12T18:10:47Z</td>
<td>2019-11-23T06:36:54Z</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>location_uri</td>
<td>Object or file URI. The format of the URI depends on the Cloud Object Storage service you are using, for details see DBMS_CLOUD Package File URI Formats.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Object Store</th>
<th>CREATED</th>
<th>LAST_MODIFIED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Cloud Infrastructure Native</td>
<td>Returns timestamp</td>
<td>Returns <code>NULL</code></td>
</tr>
<tr>
<td>Oracle Cloud Infrastructure Swift</td>
<td>Returns <code>NULL</code></td>
<td>Returns timestamp</td>
</tr>
<tr>
<td>Oracle Cloud Infrastructure Classic</td>
<td>Returns <code>NULL</code></td>
<td>Returns timestamp</td>
</tr>
<tr>
<td>Amazon S3</td>
<td>Returns <code>NULL</code></td>
<td>Returns timestamp</td>
</tr>
<tr>
<td>Azure</td>
<td>Returns timestamp</td>
<td>Returns timestamp</td>
</tr>
</tbody>
</table>

- The checksum value is the MD5 checksum. This is a 32-character hexadecimal number that is computed on the object contents.

- 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:

```sql
SELECT * FROM DBMS_CLOUD.LIST_OBJECTS('OBJ_STORE_CRED',
  'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/');
```

<table>
<thead>
<tr>
<th>OBJECT_NAME</th>
<th>BYTES</th>
<th>CHECKSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>cwallet.sso</td>
<td>2965</td>
<td>2339a2731ba24a837b26d344d643dc07</td>
</tr>
<tr>
<td></td>
<td>2019-11-23T06:36:54Z</td>
<td></td>
</tr>
</tbody>
</table>

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.

PUT_OBJECT Procedure

This procedure copies a file from Autonomous Database to the Cloud Object Storage. The maximum file size allowed in this procedure is 5 gigabytes (GB).
**Syntax**

```
DBMS_CLOUD.PUT_OBJECT (  
    credential_name      IN VARCHAR2,  
    object_uri           IN VARCHAR2,  
    directory_name       IN VARCHAR2,  
    file_name            IN VARCHAR2);  
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>object_uri</td>
<td>Object or file URI. The format of the URI depends on the Cloud Object Storage service you are using, for details see DBMS_CLOUD Package File URI Formats.</td>
</tr>
<tr>
<td>directory_name</td>
<td>The name of the directory on the Autonomous Data Warehouse database.</td>
</tr>
<tr>
<td>file_name</td>
<td>The name of the file in the specified directory.</td>
</tr>
</tbody>
</table>

**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 `adwc_user`:

```
GRANT READ ON DIRECTORY data_pump_dir TO adwc_user;
```

**Usage Note**

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 Package File URI Formats for more information.

**UPDATE_CREDENTIAL Procedure**

This procedure updates cloud service credential attributes in Autonomous Database.

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. This procedure lets you update an attribute with a new value for a specified `credential_name`. 

---

**Appendix A**
Summary of DBMS_CLOUD Subprograms
Syntax

```sql
DBMS_CLOUD.UPDATE_CREDENTIAL (  
    credential_name   IN VARCHAR2,  
    attribute         IN VARCHAR2,  
    value             IN VARCHAR2);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>The name of the credential to be stored.</td>
</tr>
<tr>
<td>attribute</td>
<td>Name of attribute to update: USERNAME or PASSWORD.</td>
</tr>
<tr>
<td>value</td>
<td>New value for the selected attribute.</td>
</tr>
</tbody>
</table>

Usage Notes

- The user name is case sensitive. It cannot contain double quotes or spaces.
- 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.

Example

```sql
BEGIN  
    DBMS_CLOUD.UPDATE_CREDENTIAL(  
        credential_name => 'OBJ_STORE_CRED',  
        attribute => 'PASSWORD',  
        value => 'password');  
END;
/```

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

```sql
DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE (  
    table_name      IN VARCHAR2,  
    schema_name     IN VARCHAR2 DEFAULT,  
    rowcount        IN NUMBER DEFAULT,  
    stop_on_error   IN BOOLEAN DEFAULT);
```

```sql
DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE(
```
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>The name of the external table.</td>
</tr>
<tr>
<td>operation_id</td>
<td>Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.</td>
</tr>
<tr>
<td>schema_name</td>
<td>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.</td>
</tr>
<tr>
<td>rowcount</td>
<td>Number of rows to be scanned. The default value is NULL meaning all the rows in the source files are scanned.</td>
</tr>
<tr>
<td>stop_on_error</td>
<td>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.</td>
</tr>
</tbody>
</table>

Usage Note

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_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,  
  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,
  schema_name IN VARCHAR2 DEFAULT,
  rowcount IN NUMBER DEFAULT,
  partition_key_validation IN BOOLEAN DEFAULT,
  stop_on_error IN BOOLEAN DEFAULT);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>The name of the external table.</td>
</tr>
<tr>
<td>operation_id</td>
<td>Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.</td>
</tr>
<tr>
<td>partition_name</td>
<td>If defined, then only a specific partition is validated. If not specified then read all partitions sequentially until rowcount is reached.</td>
</tr>
<tr>
<td>schema_name</td>
<td>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.</td>
</tr>
<tr>
<td>rowcount</td>
<td>Number of rows to be scanned. The default value is NULL meaning all the rows in the source files are scanned.</td>
</tr>
<tr>
<td>partition_key_validation</td>
<td>For internal use only. Do not use this parameter.</td>
</tr>
<tr>
<td>stop_on_error</td>
<td>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.</td>
</tr>
</tbody>
</table>

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. Hybrid partitioned tables are only supported with Oracle Database 19c onwards.

Syntax

DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE(
  table_name IN VARCHAR2,
  partition_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,  
  schema_name IN VARCHAR2 DEFAULT,  
  rowcount IN NUMBER DEFAULT,  
  partition_key_validation IN BOOLEAN DEFAULT,  
  stop_on_error IN BOOLEAN DEFAULT);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>The name of the external table.</td>
</tr>
<tr>
<td>operation_id</td>
<td>Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.</td>
</tr>
<tr>
<td>partition_name</td>
<td>If defined, then only a specific partition is validated. If not specified then read from all external partitions sequentially until rowcount is reached.</td>
</tr>
<tr>
<td>schema_name</td>
<td>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.</td>
</tr>
<tr>
<td>rowcount</td>
<td>Number of rows to be scanned. The default value is NULL meaning all the rows in the source files are scanned.</td>
</tr>
<tr>
<td>partition_key_validation</td>
<td>For internal use only. Do not use this parameter.</td>
</tr>
<tr>
<td>stop_on_error</td>
<td>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.</td>
</tr>
</tbody>
</table>

Usage Note

Hybrid partitioned tables are only supported with Oracle Database 19c onwards.

DBMS_CLOUD Package File 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

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. From the Oracle Cloud Infrastructure left navigation list click Object Storage → Object Storage.
2. From the Name column, select a bucket.
3. In the Objects area, click View Object Details.
4. 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.
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-uDtWNsiD910UCYGue/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:

```sql
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;
/
```

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`:

```sql
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 adwc bucket in the us-west-2 region.

https://s3-us-west-2.amazonaws.com/adwc/channels.txt

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 adwc container in the storage account adwc_user:

https://adwc_user.blob.core.windows.net/adwc/channels.txt

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')

Note:

For Parquet and Avro format options, see DBMS_CLOUD Package Format Options for Parquet and Avro.

As noted in the Format Option column with the text, "Option valid with JSON data", the following format options are valid for a format argument supplied with the DBMS_CLOUD.COPY_COLLECTION procedure: characterset, compression, ignoreblanklines, recorddelimiter, rejectlimit, unpackarrays. For the DBMS_CLOUD.COPY_COLLECTION procedure other listed format options are not supported.
<table>
<thead>
<tr>
<th>Format Option</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>blankasnull</td>
<td>When set to <code>true</code>, loads fields consisting of spaces as null.</td>
<td><code>blankasnull : true</code> Default value: <code>False</code></td>
</tr>
<tr>
<td>characterset</td>
<td>Specifies the characterset of source files</td>
<td><code>characterset : string</code> Default value: Database characterset</td>
</tr>
<tr>
<td>compression</td>
<td>Specifies the compression type of the source file. ZIP archiving format is not supported. Specifying the value <code>auto</code> checks for the compression types: <code>gzip</code>, <code>zlib</code>, <code>bzip2</code>.</td>
<td>`compression : auto</td>
</tr>
<tr>
<td>conversionerrors</td>
<td>If a row is rejected because of data type conversion errors, the related columns are stored as null or the row is rejected.</td>
<td>`conversionerrors : reject_record</td>
</tr>
<tr>
<td>dateformat</td>
<td>Specifies the date format in the source file. The format option <code>AUTO</code> searches for the following formats:</td>
<td><code>dateformat : string</code> Default value: Database date format</td>
</tr>
<tr>
<td></td>
<td><code>J</code> <code>MM-DD-YYYYBC</code> <code>MM-DD-YYYY</code> <code>YYYYMMDD HHMISS</code> <code>YNNMDD HHMISS</code> <code>YYYY.DDD</code> <code>YYYY-MM-DD</code></td>
<td></td>
</tr>
<tr>
<td>delimiter</td>
<td>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:</td>
<td><code>delimiter : character</code> Default value: <code>(pipe character)</code></td>
</tr>
<tr>
<td></td>
<td>format =&gt; <code>json_object('delimiter' value 'X'&quot;9&quot;')</code></td>
<td></td>
</tr>
<tr>
<td>escape</td>
<td>The character &quot;&quot; is used as the escape character when specified.</td>
<td><code>escape : true</code> Default value: <code>False</code></td>
</tr>
<tr>
<td>ignoreblanklines</td>
<td>Blank lines are ignored when set to <code>true</code>.</td>
<td><code>ignoreblanklines : true</code> Default value: <code>False</code></td>
</tr>
<tr>
<td>Ignoremissingcolumns</td>
<td>If there are more columns in the <code>field_list</code> than there are in the source files, the extra columns are stored as null.</td>
<td><code>ignoremissingcolumns : true</code> Default value: <code>False</code></td>
</tr>
<tr>
<td>language</td>
<td>Specifies a language name (for example, <code>FRENCH</code>), from which locale-sensitive information can be derived.</td>
<td><code>language : string</code> Default value: <code>Null</code> See <code>Locale Data in Oracle Database Globalization Support Guide</code> for a listing of Oracle-supported languages.</td>
</tr>
<tr>
<td>Format Option</td>
<td>Description</td>
<td>Syntax</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>numericcharacters</td>
<td>Specifies the characters to use as the group separator and decimal character.</td>
<td>numericcharacters: 'decimal_character group_separator'</td>
</tr>
<tr>
<td></td>
<td><em>decimal_character</em>: The decimal separates the integer portion of a number from the decimal portion.</td>
<td>Default value: &quot;,&quot;</td>
</tr>
<tr>
<td></td>
<td><em>group_separator</em>: The group separator separates integer groups (that is, thousands, millions, billions, and so on).</td>
<td>See NLS_NUMERIC_CHARACTERS in Oracle Database Globalization Support Guide for more information.</td>
</tr>
<tr>
<td>numberformat</td>
<td>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.</td>
<td>numberformat: number_format_model</td>
</tr>
<tr>
<td></td>
<td>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 &quot;\r\n&quot;. If it finds the Windows newline character, this is used as the record delimiter for the file. If a Windows newline character is not found, it searches for the UNIX/Linux newline character &quot;\n&quot; and if it finds one it uses it as the record delimiter for the file. Specify this argument explicitly if you want to override the default behavior, for example:</td>
<td>Default value: is derived from the setting of the NLS_TERRITORY parameter See Number Format Models in SQL Language Reference for more information.</td>
</tr>
<tr>
<td>quote</td>
<td>Specifies the quote character for the fields, the quote characters are removed during loading when specified.</td>
<td>quote: character</td>
</tr>
<tr>
<td></td>
<td>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 &quot;\r\n&quot;. If it finds the Windows newline character, this is used as the record delimiter for the file. If a Windows newline character is not found, it searches for the UNIX/Linux newline character &quot;\n&quot; and if it finds one it uses it as the record delimiter for the file. Specify this argument explicitly if you want to override the default behavior, for example:</td>
<td>Default value: Null meaning no quote</td>
</tr>
<tr>
<td>recorddelimiter</td>
<td>Specifies the record delimiter.</td>
<td>recorddelimiter: character</td>
</tr>
<tr>
<td></td>
<td>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 &quot;\r\n&quot;. If it finds the Windows newline character, this is used as the record delimiter for the file. If a Windows newline character is not found, it searches for the UNIX/Linux newline character &quot;\n&quot; and if it finds one it uses it as the record delimiter for the file. Specify this argument explicitly if you want to override the default behavior, for example:</td>
<td>Default value: newline</td>
</tr>
<tr>
<td>rejectlimit</td>
<td>The operation will error out after specified number of rows are rejected.</td>
<td>rejectlimit: number</td>
</tr>
<tr>
<td></td>
<td>Default value: 0</td>
<td></td>
</tr>
<tr>
<td>removequotes</td>
<td>Removes any quotes that are around any field in the source file.</td>
<td>removequotes: true</td>
</tr>
<tr>
<td></td>
<td>Default value: False</td>
<td></td>
</tr>
<tr>
<td>skipheaders</td>
<td>Specifies how many rows should be skipped from the start of the file.</td>
<td>skipheaders: number</td>
</tr>
<tr>
<td></td>
<td>Default value: 0 if not specified, 1 if specified without a value</td>
<td></td>
</tr>
<tr>
<td>territory</td>
<td>Specifies a territory name to further determine input data characteristics.</td>
<td>territory: string</td>
</tr>
<tr>
<td></td>
<td>Default value: Null</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format Option</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestampformat</td>
<td>Specifies the timestamp format in the source file. The format option AUTO searches for the following formats:</td>
<td>timestampformat: <code>string</code></td>
</tr>
<tr>
<td></td>
<td>YYYY-MM-DD HH:MI:SS.FF</td>
<td>Default value: Database timestamp format</td>
</tr>
<tr>
<td></td>
<td>YYYY-MM-DD HH:MI:SS.FF3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM/DD/YYYY HH:MI:SS.FF</td>
<td></td>
</tr>
<tr>
<td>timestampfmtzform</td>
<td>Specifies the timestamp with local timezone format in the source file. The format option AUTO searches for the following formats:</td>
<td>timestampfmtzform: <code>string</code></td>
</tr>
<tr>
<td></td>
<td>DD Mon YYYY HH:MI:SS F ZR</td>
<td>Default value: Database timestamp with local timezone format</td>
</tr>
<tr>
<td></td>
<td>MM/DD/YYYY HH:MI:SS F ZR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYY-MM-DD HH:MI:SS+/-TZR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYY-MM-DD HH:MI:SS.FF3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DD.MM.YYYY HH:MI:SS ZR</td>
<td></td>
</tr>
<tr>
<td>timestamptzformat</td>
<td>Specifies the timestamp with timezone format in the source file. The format option AUTO searches for the following formats:</td>
<td>timestamptzformat: <code>string</code></td>
</tr>
<tr>
<td></td>
<td>DD Mon YYYY HH:MI:SS F ZR</td>
<td>Default value: Database timestamp with timezone format</td>
</tr>
<tr>
<td></td>
<td>MM/DD/YYYY HH:MI:SS F ZR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYY-MM-DD HH:MI:SS+/-TZR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYY-MM-DD HH:MI:SS.FF3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DD.MM.YYYY HH:MI:SS ZR</td>
<td></td>
</tr>
<tr>
<td>trimspaces</td>
<td>Specifies how the leading and trailing spaces of the fields are trimmed.</td>
<td>trimspaces: rtrim</td>
</tr>
<tr>
<td></td>
<td>See the description of trim_spec.</td>
<td>Default value: notrim</td>
</tr>
<tr>
<td>truncatecol</td>
<td>If the data in the file is too long for a field, this option will truncate the value of the field rather than reject the row.</td>
<td>truncatecol: true</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the source file type.</td>
<td>type: csv</td>
</tr>
<tr>
<td></td>
<td>See the description of CSV in field_definitions Clause</td>
<td>Default value: Null</td>
</tr>
<tr>
<td></td>
<td>If the type is datapump, then the only other valid format option is rejectlimit.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>See DBMS_CLOUD Package Format Options for Parquet and Avro for type values parquet and avro.</td>
<td></td>
</tr>
<tr>
<td>unpackarrays</td>
<td>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 is valid only for JSON collection data.</td>
<td>unpackarrays: true</td>
</tr>
</tbody>
</table>

This option is valid only with JSON data.
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')

<table>
<thead>
<tr>
<th>Format Option</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Specifies the file type.</td>
<td>type : parquet</td>
</tr>
<tr>
<td>Format Option</td>
<td>Description</td>
<td>Syntax</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td>schema</td>
<td>When schema is set to first or all, the external table columns and data types are automatically derived from the Parquet or Avro file. The column names will match those found in Parquet or Avro. The data types are converted from Parquet or Avro data types to Oracle data types. All columns are added to the table. Specifies to use the metadata from the first Parquet or Avro file or from all Parquet or Avro files 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. Use all (slower) if the files may have different schemas.</td>
<td>schema : first</td>
</tr>
</tbody>
</table>

**Note:**
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.

---

### DBMS_CLOUD Package Parquet to Oracle Data Type Mapping

Describes the mapping of Parquet data types to Oracle data types.

**Note:**
The external table supports scalar data types only. Complex types, such as maps, arrays, and structs, are not supported.

<table>
<thead>
<tr>
<th>Parquet Type</th>
<th>Oracle Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>UINT_64</td>
<td>NUMBER(20)</td>
</tr>
<tr>
<td>INT_64</td>
<td>NUMBER(19)</td>
</tr>
<tr>
<td>UINT_32</td>
<td>NUMBER(10)</td>
</tr>
<tr>
<td>INT_32</td>
<td>NUMBER(10)</td>
</tr>
<tr>
<td>UINT_16</td>
<td>NUMBER(5)</td>
</tr>
<tr>
<td>INT_16</td>
<td>NUMBER(5)</td>
</tr>
<tr>
<td>Parquet Type</td>
<td>Oracle Type</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>UINT_8</td>
<td>NUMBER(3)</td>
</tr>
<tr>
<td>INT_8</td>
<td>NUMBER(3)</td>
</tr>
<tr>
<td>BOOL</td>
<td>NUMBER(1)</td>
</tr>
<tr>
<td>UTF8_BYTE_ARRAY</td>
<td>VARCHAR2(4000 BYTE)</td>
</tr>
<tr>
<td>FLT</td>
<td>BINARY_FLOAT</td>
</tr>
<tr>
<td>DBL</td>
<td>BINARY_DOUBLE</td>
</tr>
<tr>
<td>DECIMAL(p)</td>
<td>NUMBER(p)</td>
</tr>
<tr>
<td>DECIMAL(p,s)</td>
<td>NUMBER(p,s)</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>STRING</td>
<td>VARCHAR2(4000)</td>
</tr>
<tr>
<td>TIME_MILLIS</td>
<td>VARCHAR2(20 BYTE)</td>
</tr>
<tr>
<td>TIME_MILLISUtc</td>
<td>VARCHAR2(20 BYTE)</td>
</tr>
<tr>
<td>TIME_MICROS</td>
<td>VARCHAR2(20 BYTE)</td>
</tr>
<tr>
<td>TIME_MICROSUtc</td>
<td>VARCHAR2(20 BYTE)</td>
</tr>
<tr>
<td>TIMESTAMP_MILLIS</td>
<td>TIMESTAMP(3)</td>
</tr>
<tr>
<td>TIMESTAMP_MILLISUtc</td>
<td>TIMESTAMP(3)</td>
</tr>
<tr>
<td>TIMESTAMP_MICROS</td>
<td>TIMESTAMP(6)</td>
</tr>
<tr>
<td>TIMESTAMP_MICROSUtc</td>
<td>TIMESTAMP(6)</td>
</tr>
<tr>
<td>TIMESTAMP_NANOS</td>
<td>TIMESTAMP(9)</td>
</tr>
</tbody>
</table>
DBMS_CLOUD Package Avro to Oracle Data Type Mapping

Describes the mapping of Avro data types to Oracle data types.

**Note:**
For Avro, the external table supports scalar data types only, with the following exceptions:

- **DBMS_CLOUD supports** `UNION` of types `[null, SIMPLE_TYPE]`, where `SIMPLE_TYPE` is: INT, LONG, FLOAT, DOUBLE, STRING, BYTES.
- **DBMS_CLOUD does not support** `UNION` of multiple types, for example `[null, INT, DOUBLE]`
- Files that contain arrays and/or maps of simple types can be read. These columns are skipped.
- Files that contain arrays and/or maps of complex types, for example an array of records, cannot be read and report an error. For example: querying an Avro file with unsupported types shows:

  ORA-29913: error in executing ODCIEXPORTTABLEOPEN
  calloutdbms_cloud.create_external_table
  threw an error when attempting to automatically define columns that aren't supported:
  ORA-20000: Error building column list from file

- Files that contain records of simple types can be read.

<table>
<thead>
<tr>
<th>Avro Type</th>
<th>Oracle Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>NUMBER(10)</td>
</tr>
<tr>
<td>LONG</td>
<td>NUMBER(19)</td>
</tr>
<tr>
<td>BOOL</td>
<td>NUMBER(1)</td>
</tr>
<tr>
<td>UTF8 BYTE_ARRAY</td>
<td>RAW(2000)</td>
</tr>
<tr>
<td>FLT</td>
<td>BINARY_FLOAT</td>
</tr>
<tr>
<td>DBL</td>
<td>BINARY_DOUBLE</td>
</tr>
<tr>
<td>DECIMAL(p)</td>
<td>NUMBER(p)</td>
</tr>
<tr>
<td>DECIMAL(p,s)</td>
<td>NUMBER(p,s)</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>STRING</td>
<td>VARCHAR2</td>
</tr>
<tr>
<td>TIME_MILLIS</td>
<td>VARCHAR2(20 BYTE)</td>
</tr>
<tr>
<td>TIME_MICROS</td>
<td>VARCHAR2(20 BYTE)</td>
</tr>
<tr>
<td>TIMESTAMP_MILLIS</td>
<td>TIMESTAMP(3)</td>
</tr>
<tr>
<td>TIMESTAMP_MICROS</td>
<td>TIMESTAMP(6)</td>
</tr>
<tr>
<td>ENUM</td>
<td>VARCHAR2(n)</td>
</tr>
<tr>
<td>DURATION</td>
<td>RAW(2000)</td>
</tr>
</tbody>
</table>
### DBMS_CLOUD Package Parquet and AVRO to Oracle Column Name Mapping

Describes rules for how Parquet and AVRO column names are converted to Oracle column names.

The following are supported for Parquet and AVRO 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 Parquet or AVRO column names:

- Embedded blanks
- Leading numbers
- Leading underscores
- Oracle SQL reserved words

The following table shows various types of Parquet and AVRO column names, and rules for using the column names in Oracle column names in external tables.

<table>
<thead>
<tr>
<th>Parquet or AVRO name</th>
<th>CREATE TABLE Name</th>
<th>Oracle CATALOG</th>
<th>Valid SQL</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>part, Part, or PART</td>
<td>part, Part, PART</td>
<td>PART</td>
<td>select part, select Part, select paRt, select PART</td>
<td>Oracle implicitly uppercases unquoted column names</td>
</tr>
<tr>
<td>Ord No</td>
<td>&quot;Ord No&quot;</td>
<td>Ord No</td>
<td>select &quot;Ord No&quot;</td>
<td>Double quotes are required when there are embedded blanks, which also preserves the character case</td>
</tr>
<tr>
<td><strong>index_key</strong></td>
<td>&quot;<strong>index_key</strong>&quot;</td>
<td><strong>index_key</strong></td>
<td>select &quot;<strong>index_key</strong>&quot;</td>
<td>Double quotes are required when there is a leading underscore, which also preserves the character case</td>
</tr>
<tr>
<td>Parquet or AVRO name</td>
<td>CREATE TABLE Name</td>
<td>Oracle CATALOG</td>
<td>Valid SQL</td>
<td>Notes</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------</td>
<td>----------------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>6Way</td>
<td>&quot;6Way&quot;</td>
<td>6Way</td>
<td>select &quot;6Way&quot;</td>
<td>Double quotes are required when there is a leading numeric digit, which also preserves the character case.</td>
</tr>
<tr>
<td>create, Create, or CREATE, and so on. (any case variation) partition, Partition, PARTITION, and so on (for an Oracle Reserved word)</td>
<td>&quot;CREATE&quot; &quot;PARTITION&quot;</td>
<td>CREATE PARTITION</td>
<td>select &quot;CREATE&quot; select &quot;PARTITION&quot;</td>
<td>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.</td>
</tr>
<tr>
<td>rowid, Rowid, ROWid, and so on (for ROWID see notes)</td>
<td>rowid</td>
<td>select &quot;rowid&quot; select &quot;Rowid&quot; select &quot;ROWid&quot; select &quot;rowid&quot;</td>
<td>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 &quot;rowid&quot; and must always be double-quoted when referenced.</td>
<td></td>
</tr>
</tbody>
</table>
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 or Parquet column name.
- After using `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` to create an external table with AVRO or Parquet format specified, use the `DESCRIBE` command in SQL*Plus to view the table's column names.
- When Oracle SQL Reserved Words are used in Parquet or AVRO column names, they must always be double-quoted when referenced anywhere in SQL. See Oracle SQL Reserved Words for more information.

Summary of DBMS_CLOUD_ADMIN Subprograms

This section covers the `DBMS_CLOUD_ADMIN` subprograms provided with Autonomous Database.

Topics

- `CREATE_DATABASE_LINK` Procedure
- `DISABLE_APP_CONT` Procedure
- `DROP_DATABASE_LINK` Procedure
- `ENABLE_APP_CONT` Procedure
- `GRANT_TABLESPACE_QUOTA` Procedure

CREATE_DATABASE_LINK Procedure

This procedure creates a database link to a target database in the schema calling the API. You first need to upload the wallet (`cwallet.sso`) containing the certificates for the target database using `DBMS_CLOUD.GET_OBJECT` and then create the database link using the wallet.

Syntax

```sql
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,
)```

Appendix A
Summary of DBMS_CLOUD_ADMIN Subprograms
credential_name      IN VARCHAR2,
directory_name       IN VARCHAR2 DEFAULT);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>db_link_name</td>
<td>The name of the database link to create.</td>
</tr>
<tr>
<td>hostname</td>
<td>The hostname for the target database.</td>
</tr>
<tr>
<td>port</td>
<td>The port for the target database. To ensure security, ports are restricted to: 1521-1525.</td>
</tr>
</tbody>
</table>
| service_name    | The service_name for the database to link to. For a target Autonomous Database the service name consists of three parts:  
|                 | • database_name: the name of your database.                                |
|                 | • priority is one of: _high | _medium| _low  
|                 | • adwc.oraclecloud.com                                                        |
|                 | You can find the service names in the tnsnames.ora file in the wallet.zip that you download from the Autonomous Database for your connection. |
| ssl_server_cert_dn | The DN value found in the server certificate.                              |
| 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 stored cwallet.sso file. The default value for this parameter is 'data_pump_dir'. |

Usage Notes

- Autonomous Database supports creating database links only if the database is accessible through a public IP or public hostname.
- 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 adwc_user:

  GRANT EXECUTE ON DBMS_CLOUD_ADMIN TO adwc_user;
  GRANT CREATE DATABASE LINK TO adwc_user;

- 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
Example

BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DB_LINK_CRED',
    username => 'adwc_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.adwc.example.oraclecloud.com',
    ssl_server_cert_dn => 'CN=adwc.example.oraclecloud.com,OU=Oracle BMCS
FRANKFURT,O=Oracle Corporation,L=Redwood City,ST=California,C=US',
    credential_name => 'DB_LINK_CRED');
END;
/

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service_name</td>
<td>The service_name for the Autonomous Database service. The service name consists of three parts:</td>
</tr>
<tr>
<td></td>
<td>• database_name: the name of your database.</td>
</tr>
<tr>
<td></td>
<td>• priority is one of: _high</td>
</tr>
<tr>
<td></td>
<td>• adwc.oraclecloud.com</td>
</tr>
<tr>
<td></td>
<td>You can find the service names in the tnsnames.ora file in the wallet.zip that you download from an Autonomous Database for your connection.</td>
</tr>
</tbody>
</table>

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.adwc.oraclecloud.com' );
END;
/
Verify the value as follows:

```
SELECT name, failover_restore, drain_timeout FROM v$services;
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>FAILOVER_RESTORE</th>
<th>DRAIN_TIMEOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvl23abc1_adb1_high.adwc.oraclecloud.com</td>
<td>NONE</td>
<td>0</td>
</tr>
</tbody>
</table>

**DROP_DATABASE_LINK Procedure**

This procedure drops a database link.

**Syntax**

```
DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK(
    db_link_name      IN VARCHAR2);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>db_link_name</td>
<td>The name of the database link to drop.</td>
</tr>
</tbody>
</table>

**Example**

```sql
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service_name</td>
<td>The service_name for the Autonomous Database service. The service name consists of three parts:</td>
</tr>
<tr>
<td></td>
<td>• database_name: the name of your database.</td>
</tr>
<tr>
<td></td>
<td>• priority is one of: _high</td>
</tr>
<tr>
<td></td>
<td>• adwc.oraclecloud.com</td>
</tr>
<tr>
<td></td>
<td>You can find the service names in the tnsnames.ora file in the wallet.zip that you download from an Autonomous Database for your connection.</td>
</tr>
</tbody>
</table>

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.adwc.oraclecloud.com'
    );
END;
/

Verify the value as follows:

SELECT name, failover_restore, drain_timeout FROM v$services;

<table>
<thead>
<tr>
<th>NAME</th>
<th>FAILOVER_RESTORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAIN_TIMEOUT</td>
<td>-----------------</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>nvthp2ht_adb1_high.adwc.oraclecloud.com</td>
<td>LEVEL1</td>
</tr>
<tr>
<td></td>
<td>300</td>
</tr>
</tbody>
</table>

GRANT_TABLESPACE_QUOTA Procedure

This procedure grants a storage quota to a specified database user. When a tablespace quota is granted to a user, Autonomous Database limits the storage space used by that user to the specified quota. Using the value UNLIMITED specifies unlimited tablespace privilege.
Syntax

```sql
DBMS_CLOUD_ADMIN.GRANT_TABLESPACE_QUOTA(
  username            IN VARCHAR2,
  tablespace_quota    IN VARCHAR2);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_name</td>
<td>The database username to grant the tablespace quota to.</td>
</tr>
<tr>
<td>tablespace_quota</td>
<td>The quota to assign to the specified user in bytes. For kilobytes,</td>
</tr>
<tr>
<td></td>
<td>megabytes, gigabytes, and terabytes you can specify K, M, G, or T after</td>
</tr>
<tr>
<td></td>
<td>the numeric value respectively. Alternatively you can specify UNLIMITED,</td>
</tr>
<tr>
<td></td>
<td>which is equivalent to GRANT UNLIMITED TABLESPACE with SQL.</td>
</tr>
</tbody>
</table>

Usage Notes

See System Privileges (Organized by the Database Object Operated Upon - TABLESPACE) for more information.

See Manage User Privileges with Autonomous Data Warehouse for information on privileges granted with the role DWROLE.

Examples

```sql
BEGIN
  DBMS_CLOUD_ADMIN.GRANT_TABLESPACE_QUOTA(
    username => 'ADBUSER',
    tablespace_quota => '10G'
  );
END;
/

BEGIN
  DBMS_CLOUD_ADMIN.GRANT_TABLESPACE_QUOTA(
    username => 'ADBUSER',
    tablespace_quota => 'UNLIMITED'
  );
END;
/```

Summary of DBMS_MAX_STRING_SIZE Subprograms

The `DBMS_MAX_STRING_SIZE` package provides an interface for checking and changing the value of the `DBMS_MAX_STRING_SIZE` initialization parameter.

Topics

- CHECK_MAX_STRING_SIZE Function
• MODIFY_MAX_STRING_SIZE Procedure

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>new_value</td>
<td>Specifies the new MAX_STRING_SIZE parameter value to be set. The only valid value is: 'STANDARD'.</td>
</tr>
</tbody>
</table>

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'));
```

<table>
<thead>
<tr>
<th>TYPE</th>
<th>OBJECT_OWNER</th>
<th>OBJECT_NAME</th>
<th>COLUMN_NAME</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>CUST_NOTES</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>new_value</td>
<td>Specifies the new MAX_STRING_SIZE parameter value to be set. The only valid value is: 'STANDARD'.</td>
</tr>
</tbody>
</table>

### Usage Notes

- 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

```sql
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.

```sql
SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME = 'max_string_size';

NAME     VALUE
max_string_size  STANDARD
```

### Summary of DBMS_CLOUD_MACADM Subprograms

This section covers the DBMS_CLOUD_MACADM subprograms provided with Autonomous Database.

### Topics

- CONFIGURE_DATABASE_VAULT Procedure
- DISABLE_DATABASE_VAULT Procedure
- DISABLE_USERMGMT_DATABASE_VAULT Procedure
- ENABLE_DATABASE_VAULT Procedure
- ENABLE_USERMGMT_DATABASE_VAULT Procedure
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dvowner_uname</td>
<td>Name of the user who will be the Database Vault Owner. This user will be granted the DV_OWNER role.</td>
</tr>
<tr>
<td>dvacctmgr_uname</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

```sql
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

```sql
DBMS_CLOUD_MACADM.DISABLE_USERMGMT_DATABASE_VAULT('component_name');
```

Where: `component_name` is the component name. Valid value is: APEX.
APEX is the Oracle Application Express 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:

```sql
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

```sql
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 Application Express 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 REST APIs**

This section covers the **DBMS_CLOUD** REST APIs provided with Autonomous Data Warehouse.

---

**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 **adwc_user**:

```
GRANT EXECUTE ON DBMS_CLOUD TO adwc_user;
```
DBMS_CLOUD REST API Overview

When you use PL/SQL in your application and you need to call Cloud REST APIs you can use the `DBMS_CLOUD` function `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`. These functions provide a generic API that lets you call any REST API with the following supported cloud services:

- Oracle Cloud Infrastructure
- Amazon Web Services (AWS)
- Azure Cloud
- Oracle Cloud Infrastructure Classic

See the following for more information:

- See [API Reference and Endpoints](#) for information on Oracle Cloud Infrastructure REST APIs.
- See [Guides and API References](#) for information on Amazon Web Services REST APIs.
- See [Azure REST API Reference](#) for information on Azure REST APIs.
- See [All REST Endpoints](#) for information on Oracle Cloud Infrastructure Classic REST APIs.

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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD_DELETE</td>
<td>VARCHAR2 (6)</td>
<td>'DELETE'</td>
</tr>
<tr>
<td>METHOD_GET</td>
<td>VARCHAR2 (3)</td>
<td>'GET'</td>
</tr>
</tbody>
</table>

1 Support for Azure Cloud REST API calls is limited to the domain "blob.windows.net".
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD_HEAD</td>
<td>VARCHAR2(4)</td>
<td>'HEAD'</td>
</tr>
<tr>
<td>METHOD_POST</td>
<td>VARCHAR2(4)</td>
<td>'POST'</td>
</tr>
<tr>
<td>METHOD_PUT</td>
<td>VARCHAR2(3)</td>
<td>'PUT'</td>
</tr>
</tbody>
</table>

### GET_RESPONSE_HEADERS Function

This function returns the HTTP response headers as JSON data in a JSON object.

**Syntax**

```sql
DBMS_CLOUD.GET_RESPONSE_HEADERS(
    resp          IN DBMS_CLOUD_TYPES.resp
) RETURN JSON_OBJECT_T;
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resp</td>
<td>HTTP Response type returned from DBMS_CLOUD.SEND_REQUEST.</td>
</tr>
</tbody>
</table>

**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid_response</td>
<td>ORA-20025</td>
<td>Invalid response type object passed to DBMS_CLOUD.GET_RESPONSE_HEADERS.</td>
</tr>
</tbody>
</table>

### 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**

```sql
DBMS_CLOUD.GET_RESPONSE_RAW(
    resp     IN DBMS_CLOUD_TYPES.resp
) RETURN BLOB;
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resp</td>
<td>HTTP Response type returned from DBMS_CLOUD.SEND_REQUEST.</td>
</tr>
</tbody>
</table>
**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid_response</td>
<td>ORA-20025</td>
<td>Invalid response type object passed to DBMS_CLOUD.GET_RESPONSE_RAW.</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resp</td>
<td>HTTP Response type returned from DBMS_CLOUD.SEND_REQUEST.</td>
</tr>
</tbody>
</table>

**Exceptions**

<table>
<thead>
<tr>
<th>Exception</th>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid_response</td>
<td>ORA-20025</td>
<td>Invalid response type object passed to DBMS_CLOUD.GET_RESPONSE_STATUS_CODE.</td>
</tr>
</tbody>
</table>

**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 the HTTP response is in text format.

**Syntax**

```
DBMS_CLOUD.GET_RESPONSE_TEXT(
    resp          IN DBMS_CLOUD_TYPES.resp
) RETURN CLOB;
```
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resp</td>
<td>HTTP Response type returned from DBMS_CLOUD.SEND_REQUEST.</td>
</tr>
</tbody>
</table>

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid_response</td>
<td>ORA-20025</td>
<td>Invalid response type object passed to DBMS_CLOUD.GET_RESPONSE _TEXT.</td>
</tr>
</tbody>
</table>

SEND_REQUEST Function

This 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 a return response code and payload.

Syntax

```sql
DBMS_CLOUD.SEND_REQUEST(
    credential_name    IN VARCHAR2,
    uri                IN VARCHAR2,
    method             IN VARCHAR2,
    headers            IN CLOB DEFAULT NULL,
    body               IN BLOB DEFAULT NULL)
RETURN DBMS_CLOUD_TYPES.resp;
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>The name of the credential for authenticating with the corresponding cloud native API.</td>
</tr>
<tr>
<td>uri</td>
<td>HTTP URI to make the request.</td>
</tr>
<tr>
<td>method</td>
<td>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.</td>
</tr>
<tr>
<td>headers</td>
<td>HTTP Request headers for the corresponding cloud native API in JSON format. The authentication headers are set automatically, only custom headers should be passed.</td>
</tr>
<tr>
<td>body</td>
<td>HTTP Request Body for PUT and POST requests.</td>
</tr>
</tbody>
</table>
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid_req_method</td>
<td>ORA-20023</td>
<td>Request method passed to <code>DBMS_CLOUD.SEND_REQUEST</code> is invalid.</td>
</tr>
<tr>
<td>invalid_req_header</td>
<td>ORA-20024</td>
<td>Request headers passed to <code>DBMS_CLOUD.SEND_REQUEST</code> are not in valid JSON format.</td>
</tr>
</tbody>
</table>

Usage Note

If you are using Oracle Cloud Infrastructure, you must use a Signing Key based credential value for the `credential_name`. See CREATE_CREDENTIAL Procedure (OCI Signing Key Credentials) for more information.

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..aaaaaaaauq54mi7zdyfhw33ozkuontjceel17fok5nq3bf2vwe tkpqsoa',  
        tenancy_ocid    => 'ocid1.tenancy.oc1..aabbbbbbaafcue47pqmrfr4vigneegbcmmbloy5r7xvoypicj qqge32ewnrcyx2a',  
        private_key     => 'MIIEogIBAAKCAQEAtUnxbmrekgwVac6FWeRzoXvIpaA9+0r1.....wtNpEsQQ0QL GPD8NM//JEBg=',  
END;
/

See CREATE_CREDENTIAL Procedure (OCI Signing Key Credentials) 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 \texttt{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, \texttt{namespace-string} is the Oracle Cloud Infrastructure object storage namespace and \texttt{bucketname} is the bucket name. See Understanding Object Storage Namespaces for more information.
- Where: \texttt{region} is an endpoint region. See Object Storage API reference in API Reference and Endpoints for more information. For example, where \texttt{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 \texttt{bucketname}. 

See **DeleteBucket** for details on the Oracle Cloud Infrastructure Object Storage Service API for this example.

```sql
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.

```sql
--
-- 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) ||
      DBMS_CLOUD.get_response_headers(resp).to_clob || CHR(10));
   dbms_output.put_line('Status Code: ' || '------------' ||
      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.
Autonomous Data Warehouse for Experienced Oracle Database Users

This appendix provides information on using Autonomous Data Warehouse for experienced Oracle Database users with Autonomous Database on shared Exadata infrastructure.

For equivalent information about using Oracle Database features and options with Autonomous Database on dedicated Exadata infrastructure, see these resources:

- Autonomous Data Warehouse on dedicated Exadata infrastructure: see Using Oracle Database Features in Dedicated Autonomous Database Deployments.
- Autonomous Transaction Processing on dedicated Exadata infrastructure: see Using Oracle Database Features in Dedicated Autonomous Database Deployments.

Topics

- About the Autonomous Data Warehouse Database
- Autonomous Data Warehouse Oracle Database 19c Features
- Restrictions for Database Initialization Parameters
- Restrictions for SQL Commands
- Restrictions for Data Types
- Managing Partitions, Indexes, and Materialized Views
- Restrictions and Notes for Database PL/SQL Packages
- Restrictions for Database Features

About the Autonomous Data Warehouse Database

Autonomous Data Warehouse 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.

Note:

Some features of Autonomous Data Warehouse are only available with the latest database release, Oracle Database 19c.

You may be using Autonomous Data Warehouse with Oracle Database 18c or Oracle Database 19c. The version you are using depends either on the region where you provision your database, or in regions where both releases are available, your selection of an Oracle Database release when you provision your database (you can also create a database by cloning).

See Oracle Database Versions and Availability by Region for more information.

Characteristics of an Autonomous Data Warehouse database:

- 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.
- Compression is enabled by default. Autonomous Data Warehouse 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. Changing the result cache mode is not allowed.

Accessing an Autonomous Data Warehouse 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:

- Parallelism is enabled by default. Degree of parallelism for SQL statements is set based on the number of OCPUs in the system and the database service the user is connecting to. For more information on database services see Managing Concurrency and Priorities on Autonomous Data Warehouse.
- Parallel DML is enabled by default. If you do not want to run DML operations in parallel you can disable parallel DML in your session using the following SQL command:

```
ALTER SESSION DISABLE PARALLEL DML;
```
See VLDB and Partitioning Guide for more information on parallel DML operations.

Autonomous Data Warehouse Oracle Database 19c Features

If you are using Autonomous Data Warehouse with Oracle Database 19c, then there are additional database features available in the database.

Autonomous Data Warehouse with Oracle Database 19c 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.

- 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.

Restrictions for Database Initialization Parameters

Autonomous Data Warehouse 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
Restrictions for SQL Commands

Autonomous Data Warehouse allows most of the SQL commands available in the Oracle Database. To ensure the security and the performance of Autonomous Data Warehouse, some SQL commands are restricted.

This section provides a list of SQL command limitations that are required to protect security and for performance integrity in Autonomous Data Warehouse. Most of the standard SQL and PL/SQL syntax and constructs used with the Oracle Database work in Autonomous Data Warehouse.
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 Data Warehouse.

The following SQL statements are not available in Autonomous Data Warehouse:

• ADMINISTER KEY MANAGEMENT
• ALTER PROFILE
• ALTER TABLESPACE
• CREATE DATABASE LINK

Note:

Use DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK to create database links in Autonomous Data Warehouse. See Create Database Links from Autonomous Database to Other Databases for more information.

• CREATE PROFILE
• CREATE TABLESPACE
• DROP TABLESPACE

SQL Statements with Restrictions in Autonomous Data Warehouse

The following DDL statements are available in Autonomous Data Warehouse with restrictions:

<table>
<thead>
<tr>
<th>SQL Command</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTER PLUGGABLE DATABASE and ALTER DATABASE</td>
<td>Only the following clauses are allowed: DATAFILE AUTOEXTEND ON DATAFILE AUTOEXTEND OFF DATAFILE RESIZE DEFAULT EDITION SET TIME_ZONE SET CMU_WALLET</td>
</tr>
</tbody>
</table>
### SQL Command Restrictions

<table>
<thead>
<tr>
<th>SQL Command</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ALTER SESSION</strong></td>
<td>Only the following clauses are allowed:</td>
</tr>
<tr>
<td></td>
<td>ADVISE COMMIT, ADVISE ROLLBACK, ADVISE NOTHING</td>
</tr>
<tr>
<td></td>
<td>CLOSE DATABASE LINK</td>
</tr>
<tr>
<td></td>
<td>ENABLE COMMIT IN PROCEDURE, DISABLE COMMIT IN PROCEDURE</td>
</tr>
<tr>
<td></td>
<td>ENABLE PARALLEL &lt;QUERY</td>
</tr>
<tr>
<td></td>
<td>ENABLE RESUMABLE, DISABLE RESUMABLE</td>
</tr>
<tr>
<td></td>
<td>SET DEFAULT_COLLATION</td>
</tr>
<tr>
<td></td>
<td>SET EDITION</td>
</tr>
<tr>
<td></td>
<td>SET ISOLATION_LEVEL</td>
</tr>
<tr>
<td></td>
<td>SET ROW ARCHIVAL VISIBILITY</td>
</tr>
<tr>
<td></td>
<td>SET TIME_ZONE</td>
</tr>
<tr>
<td><strong>ALTER SYSTEM</strong></td>
<td>ALTER SYSTEM is not allowed except ALTER SYSTEM SET and</td>
</tr>
<tr>
<td></td>
<td>ALTER SYSTEM KILL SESSION. SET can only be used to set</td>
</tr>
<tr>
<td></td>
<td>parameters listed in Restrictions for Database Initialization</td>
</tr>
<tr>
<td></td>
<td>Parameters.</td>
</tr>
<tr>
<td><strong>ALTER USER</strong></td>
<td>The following clauses are ignored:</td>
</tr>
<tr>
<td></td>
<td>• DEFAULT TABLESPACE</td>
</tr>
<tr>
<td></td>
<td>• PROFILE</td>
</tr>
<tr>
<td></td>
<td>IDENTIFIED with the EXTERNALLY clause is not supported.</td>
</tr>
<tr>
<td><strong>ALTER TABLE</strong></td>
<td>For restrictions, see ALTER TABLE Restrictions.</td>
</tr>
<tr>
<td><strong>CREATE TABLE</strong></td>
<td>For restrictions, see CREATE TABLE Restrictions.</td>
</tr>
<tr>
<td><strong>CREATE USER</strong></td>
<td>The following clauses are ignored:</td>
</tr>
<tr>
<td></td>
<td>• DEFAULT TABLESPACE</td>
</tr>
<tr>
<td></td>
<td>• PROFILE</td>
</tr>
<tr>
<td></td>
<td>IDENTIFIED with the EXTERNALLY clause is not supported.</td>
</tr>
</tbody>
</table>

### CREATE TABLE Restrictions

**XMLType** tables are not allowed.

The clauses not in this list are allowed.

<table>
<thead>
<tr>
<th>Clause</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>physical_properties</td>
<td>Ignored</td>
</tr>
<tr>
<td>logging_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>innmemory_table_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>ilm_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>organization index</td>
<td>Ignored</td>
</tr>
<tr>
<td>organization external</td>
<td>Ignored</td>
</tr>
<tr>
<td>cluster</td>
<td>Ignored</td>
</tr>
<tr>
<td>LOB_storage_clause</td>
<td>Ignored</td>
</tr>
</tbody>
</table>
ALTER TABLE Restrictions

The clauses not in this list are allowed.

<table>
<thead>
<tr>
<th>Clause</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>physical_attributes_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>logging_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>inmemory_table_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>ilm_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>allocate_extent_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>deallocate_unused_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>shrink_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>alter_iot_clauses</td>
<td>Ignored</td>
</tr>
<tr>
<td>modify_LOB_storage_clause</td>
<td>Ignored</td>
</tr>
</tbody>
</table>

Restrictions for Data Types

Autonomous Data Warehouse allows most of the data types available in the Oracle Database. To ensure the security and the performance of Autonomous Data Warehouse, some data types are restricted.

The following data types are not supported or have limited support in Autonomous Data Warehouse:

- Media types are not supported (Oracle Multimedia is desupported)
- Oracle Spatial and Graph types are supported with restrictions. See Restrictions for Oracle Spatial and Graph for more information.

By default Autonomous Data Warehouse uses Hybrid Columnar Compression (HCC), and the following types are not supported for columns in tables created with HCC:

- `LONG`
- `LONG RAW`
Checking and Setting MAX_STRING_SIZE

By default Autonomous Data Warehouse 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 Autonomous Data Warehouse 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.
Managing Partitions, Indexes, and Materialized Views

Autonomous Data Warehouse allows manual creation of partitioned tables, indexes, and materialized views using the partitioning clause in the CREATE TABLE statement, the CREATE INDEX statement, and the CREATE MATERIALIZED VIEW statement respectively.

Oracle recommends that you do not manually create these structures, and leave performance optimizations to the Autonomous Data Warehouse. If you're a highly skilled Oracle Database tuning expert and decide to manually create these access structures, please be advised to test the impact of your manual tuning efforts on your full workload.

Restrictions and Notes for Database PL/SQL Packages

Lists Oracle Database PL/SQL packages with restrictions and notes in Autonomous Data Warehouse database.

Removed PL/SQL Packages

• **UTL_TCP**

PL/SQL Packages Restrictions

• **UTL_HTTP Restrictions**
  – Connections through IP addresses are not allowed.
  – Only HTTPS connections are allowed (**HTTP and HTTP_PROXY are disallowed**).
  – All web services must be secured. The only allowed ports are 443 and 8443. Your instance is preconfigured with an Oracle Wallet that contains more than 90 of the most commonly trusted root and intermediate SSL certificates. This Oracle Wallet is centrally managed and therefore you cannot consume 3rd party web services that are protected using self-signed SSL certificates.
    – The **SET_AUTHENTICATION_FROM_WALLET** procedure is disallowed.
    – The **WALLET_PATH** and **WALLET_PASSWORD** arguments for the **CREATE_REQUEST_CONTEXT**, **REQUEST**, and **REQUEST_PIECES** APIs are ignored.
    – Oracle Wallet configuration cannot be altered. All arguments for **SET_WALLET** API are ignored.
    – **UTL_HTTP** usage is audited by default. You cannot disable auditing for **UTL_HTTP**.
  • **UTL_SMTP Restrictions**
    – 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.
– UTL_SMTP usage is audited by default. You cannot disable auditing for UTL_SMTP.

• DBMS_NETWORK_ACL_ADMIN Restrictions
  – Granting ACL privileges on IP addresses is not allowed.
  – The http_proxy and use_passwords ACL privileges are not allowed.

See UTL_HTTP, UTL_SMTP, and DBMS_NETWORK_ACL_ADMIN in PL/SQL Packages and Types Reference for more information.

PL/SQL Packages Notes

• UTL_HTTP Errors
  The following table shows error messages and possible causes for these error messages when using UTL_HTTP:

<table>
<thead>
<tr>
<th>Error Message</th>
<th>Potential Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORA-12545: Connect failed because target host or object does not exist</td>
<td>Target host or object does not exist or it is private.</td>
</tr>
<tr>
<td>ORA-24247: network access denied by access control list (ACL)</td>
<td>Access control list (ACL) for the specified host could not be found.</td>
</tr>
<tr>
<td>ORA-29024: Certificate validation failure</td>
<td>Certificate of the host does not exist or is not among the supported certificates.</td>
</tr>
</tbody>
</table>

• UTL_HTTP Example
  Submit an HTTP request for the host www.example.com:

  -- Create an Access Control List for the host
  BEGIN
  DBMS_NETWORK_ACL_ADMIN.CREATE_ACL(acl => 'http.xml',
                                     description => 'HTTP ACL',
                                     principal => 'ADMIN',
                                     is_grant => true,
                                     privilege => 'connect');

  DBMS_NETWORK_ACL_ADMIN.ADD_PRIVILEGE(acl => 'http.xml',
                                        principal => 'ADMIN',
                                        is_grant => true,
                                        privilege => 'resolve');

  DBMS_NETWORK_ACL_ADMIN.ASSIGN_ACL(acl => 'http.xml',
                                     host => 'www.example.com');
  END;
  /

  -- Set Oracle Wallet location (no arguments needed)
  BEGIN
    UTL_HTTP.SET_WALLET('');
  END;
  /

  -- Submit an HTTP request
Restrictions for Database Features

Autonomous Data Warehouse is built for data warehouse workloads. In some cases, features which are present in Oracle Database Enterprise Edition are not available in Autonomous Data Warehouse. Additionally, database features designed for administration are not available.

Topics:
- Restrictions for Oracle XML DB
- Restrictions for Oracle Text
- Restrictions for Oracle Spatial and Graph
- Restrictions for Oracle Application Express
- Restrictions for Oracle Flashback
- Restrictions for Fast Application Notification (FAN)
- List of Restricted and Removed Oracle Features

Restrictions for Oracle XML DB

The following describes support for Oracle XML DB features in Autonomous Data Warehouse databases. To ensure the security and the performance of Autonomous Data Warehouse, 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 Data Warehouse using Oracle Data Pump, you need to convert to Non-Schema Binary XML prior to using Oracle Data Pump Export (expdp).

<table>
<thead>
<tr>
<th>Area</th>
<th>XML DB Feature</th>
<th>Supported in Autonomous Database</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>XML DB Protocol</td>
<td>No</td>
<td>Repository Access Using Protocols</td>
</tr>
<tr>
<td>Repository</td>
<td>XML DB Resources</td>
<td>No</td>
<td>Oracle XML DB Repository Resources</td>
</tr>
<tr>
<td>Repository</td>
<td>XML DB ACLs</td>
<td>No</td>
<td>Repository Access Control</td>
</tr>
<tr>
<td>Storage</td>
<td>XML Schema Registration</td>
<td>No</td>
<td>XML Schema Registration with Oracle XML DB</td>
</tr>
<tr>
<td>Storage</td>
<td>CLOB</td>
<td>No</td>
<td>Deprecated</td>
</tr>
<tr>
<td>Area</td>
<td>XML DB Feature</td>
<td>Supported in Autonomous Database</td>
<td>More Information</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------</td>
<td>----------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>Storage</td>
<td>Object Relational</td>
<td>No</td>
<td>XML Schema and Object-Relational XMLType</td>
</tr>
<tr>
<td>Storage</td>
<td>Binary XML</td>
<td>Yes (Non schema-based only)</td>
<td>XMLType Storage Models</td>
</tr>
<tr>
<td>Index</td>
<td>Structured XML Index</td>
<td>Yes</td>
<td>XMLIndex Structured Component</td>
</tr>
<tr>
<td>Index</td>
<td>XQuery Full Text Index</td>
<td>Yes</td>
<td>Indexing XML Data for Full-Text Queries</td>
</tr>
<tr>
<td>Index</td>
<td>Unstructured XML Index</td>
<td>No</td>
<td>XMLIndex Unstructured Component</td>
</tr>
<tr>
<td>Packages</td>
<td>XML DOM package</td>
<td>Yes</td>
<td>PL/SQL DOM API for XMLType (DBMS_XMLDOM)</td>
</tr>
<tr>
<td>Packages</td>
<td>XML Parser Package</td>
<td>Yes</td>
<td>PL/SQL Parser API for XMLType (DBMS_XMLPARSER)</td>
</tr>
<tr>
<td>Packages</td>
<td>XSL Processor</td>
<td>Yes</td>
<td>PL/SQL XSLT Processor for XMLType (DBMS_XSLPROCESSOR)</td>
</tr>
</tbody>
</table>

For details on Oracle XML DB, see Oracle XML DB Developer's Guide.

Restrictions for Oracle Text

The following describes support for Oracle Text features in Autonomous Data Warehouse databases. To ensure the security and the performance of Autonomous Data Warehouse, some Oracle Text features are restricted.

<table>
<thead>
<tr>
<th>Oracle Text Feature</th>
<th>Supported in Autonomous Database</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>All logging, and APIs which perform logging such as ctx_report.query_log_summary</td>
<td>Not Supported</td>
<td>QUERY_LOG_SUMMARY</td>
</tr>
<tr>
<td>File and URL datastore</td>
<td>Not Supported</td>
<td>Datastore Type</td>
</tr>
<tr>
<td>CREATE_INDEX with BIG_IO option</td>
<td>Not supported by default.¹</td>
<td>Improved Response Time Using the BIG_IO Option of CONTEXT Index</td>
</tr>
<tr>
<td>OPTIMIZE_INDEX in rebuild mode</td>
<td>Not supported by default.(see Footnote 1)</td>
<td>OPTIMIZE_INDEX</td>
</tr>
</tbody>
</table>

¹ This is supported if you grant the privilege to create a trigger to the user (GRANT CREATE TRIGGER). You must also disable parallel DML at the session level (ALTER SESSION DISABLE PARALLEL DML).

For details on Oracle Text, see Oracle Text Application Developer's Guide.
Restrictions for Oracle Spatial and Graph

The following describes support for Oracle Spatial and Graph features in Autonomous Data Warehouse databases. To ensure the security and the performance of Autonomous Data Warehouse, some Oracle Spatial and Graph features are restricted.

### Note:

Autonomous Database does not include Oracle Spatial and Graph 3-Dimensional geometry types and related operators, functions, or utilities.

<table>
<thead>
<tr>
<th>Spatial Graph Feature</th>
<th>Supported in Autonomous Database</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D Geometry data types</td>
<td>Yes</td>
<td>SDO_GEOMETRY Object Type</td>
</tr>
<tr>
<td>2D Spatial operators</td>
<td>Yes</td>
<td>Spatial Operators</td>
</tr>
<tr>
<td>2D Spatial functions</td>
<td>Yes</td>
<td>SDO_GEOM Package (Geometry)</td>
</tr>
<tr>
<td>Coordinate Transformation</td>
<td>Yes</td>
<td>SDO_CS Package (Coordinate System Transformation)</td>
</tr>
<tr>
<td>Spatial Utility Functions (except GML and KML conversion functions)</td>
<td>Yes</td>
<td>SDO_UTIL Package (Utility)</td>
</tr>
<tr>
<td>Spatial Aggregate Functions</td>
<td>Yes</td>
<td>Spatial Aggregate Functions</td>
</tr>
<tr>
<td>Spatial indexing and querying features</td>
<td>Yes</td>
<td>SQL Statements for Indexing Spatial Data</td>
</tr>
<tr>
<td>2D Linear Referencing System</td>
<td>Yes</td>
<td>Spatial Query</td>
</tr>
<tr>
<td>GeoJSON and JSON support for Spatial data types</td>
<td>Yes</td>
<td>JSON and GeoJSON Support in Oracle Spatial and Graph</td>
</tr>
<tr>
<td>Location Tracking Server</td>
<td>Yes</td>
<td>SDO_TRKR Package (Location Tracking)</td>
</tr>
<tr>
<td>Spatial Analysis and Mining features</td>
<td>Yes</td>
<td>Spatial Analysis and Mining</td>
</tr>
<tr>
<td>Flat Model for Point Cloud data</td>
<td>Yes</td>
<td>DO_PC_PKG.CLIP_PC_FLAT</td>
</tr>
<tr>
<td>RDF Graph</td>
<td>No</td>
<td>RDF Semantic Graph Overview</td>
</tr>
<tr>
<td>Property Graph</td>
<td>Yes</td>
<td>Spatial and Graph Property Graph Support Overview</td>
</tr>
<tr>
<td>Network Data Model</td>
<td>No</td>
<td>Network Data Model</td>
</tr>
<tr>
<td>GeoRaster</td>
<td>No</td>
<td>SDO_GEOR Package Reference</td>
</tr>
<tr>
<td>Routing Engine</td>
<td>No</td>
<td>Routing Engine</td>
</tr>
<tr>
<td>Geocoder</td>
<td>No</td>
<td>Geocoding Address Data</td>
</tr>
</tbody>
</table>
Spatial Graph Feature | Supported in Autonomous Database | More Information
--- | --- | ---
Spatial Visualization | No | Introduction to the Map Visualization Component
Topology Data Model | No | Topology Data Model Overview
Open Geospatial Consortium Web Services (WMS, WFS-T, C-SW, WCS, OpenLS) | No | SDO_WFS_PROCESS Package (WFS Processing), SDO_OLS Package (OpenLS)
GML | No | SDO_UTIL.TO_GMLGEOMETRY
Point Cloud Object Type | No | SDO_PC_PKG Package (Point Clouds)
Triangulated Irregular Network (TIN) types | No | SDO_TIN_PKG Package (TINs)

For details on Oracle Spatial and Graph, see Oracle Spatial and Graph Developer's Guide.

Restrictions for Oracle Application Express

Autonomous Data Warehouse supports Oracle Application Express. To ensure the security and the performance of Autonomous Data Warehouse, some Oracle Application Express features are restricted.

See Restrictions and Limitations for Oracle Application Express with Autonomous Data Warehouse for details.

Restrictions for 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 Data Warehouse Database.

Oracle Flashback Feature | Supported in Autonomous Database
--- | ---
DBMS_FLASHBACK | Yes except the procedure: DBMS_FLASHBACK.TRANSACTION_BACKOUT
Flashback Data Archive | No |
Flashback Drop | Yes |
Flashback Query | Yes |
Flashback Table | Yes |
Flashback Transaction | No |
Flashback Transaction Query | Yes |
Flashback Version Query | Yes |

See About Oracle Flashback Technology for information on using Flashback features.
Restrictions for Fast Application Notification (FAN)

Subscribing to Fast Application Notification (FAN) events is not supported with Autonomous Database and is not needed. The functionality provided by FAN is provided out of the box with Autonomous Database. You do not need to enable your application for FAN or subscribe to FAN events.

You can use Application Continuity to mask outages from your applications. See Enable and Disable Application Continuity for more information.

Recommended Client Settings for Common Drivers

- **ODP.Net**: The Oracle Data Provider for .NET (ODP.Net) unmanaged provider supports Application Continuity out of the box. When connecting an ODP.Net application to your Autonomous Database, do not explicitly set HA events, application continuity, or onConfig; accept the default settings.

- **JDBC (thin) driver**: When using the Universal Connection Pool (UCP), disable Fast Connection Failover. For example:

  ```java
  PoolDataSource.setFastConnectionFailoverEnabled(false)
  ```

- **Oracle Call Interface (OCI)**: Do not configure ONS servers in `oraaccess.xml`:

  ```xml
  <ons>
    <servers>
      <!-- Do not enter any values -->
    </servers>
  </ons>
  ```

  Also, do not configure the `<fan>` section:

  ```xml
  <fan>
    <!-- only possible values are "trace" or "error" -->
    <subscription_failure_action>
    </subscription_failure_action>
  </fan>
  ```

List of Restricted and Removed Oracle Features

Lists the Oracle Database features that are not available in Autonomous Data Warehouse. Additionally, database features designed for administration are not available.

**List of Removed Oracle Features**

- Oracle Real Application Testing
- Oracle OLAP
- Oracle R capabilities of Oracle Advanced Analytics
- Oracle Industry Data Models
Appendix B
Restrictions for Database Features

- Oracle Tuning Pack
- Oracle Database Lifecycle Management Pack
- Oracle Data Masking and Subsetting Pack
- Oracle Cloud Management Pack for Oracle Database
- Oracle Multimedia
- Java in DB
- Oracle Workspace Manager
C

Migrating Amazon Redshift to Autonomous Data Warehouse

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.

Topics
• Autonomous Data Warehouse Redshift Migration Overview
• Connect to Amazon Redshift
• Connect to Autonomous Data Warehouse
• Start the Cloud Migration Wizard
• Review and Finish the Amazon Redshift Migration
• Use Generated Amazon Redshift Migration Scripts
• Perform Post Migration Tasks

Autonomous Data Warehouse Redshift Migration Overview

Using SQL Developer you can migrate database files from Amazon Redshift to Autonomous Data Warehouse.

• 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 Data Warehouse 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 Data Warehouse. 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 Data Warehouse

Using SQL Developer you create a connection to Autonomous Data Warehouse

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 Data Warehouse.

3. For the AWS Redshift Migration connection, select the _low connection to your database. For more information, see Predefined Database Service Names for Autonomous Data Warehouse.

4. Add a connection to Autonomous Data Warehouse.
For more information, see Connecting with Oracle SQL Developer (18.2 or later).

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 Data Warehouse.

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 Data Warehouse.

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 Data Warehouse.
- 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 Data Warehouse 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 Data Warehouse 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 on 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 Data Warehouse. See Connect to Autonomous Data Warehouse. 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 Data Warehouse 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 Data Warehouse 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 Data Warehouse 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 Data Warehouse 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 Data Warehouse. 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 Data Warehouse.

**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 Data Warehouse. 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 Data Warehouse. 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 Data Warehouse, the migration process excludes deploying these selected schemas and displays a dialog:
Summary: What The Migration Assistant Creates

- Creates a new Autonomous Data Warehouse 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 Data Warehouse. 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 Data Warehouse 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 Data Warehouse 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 Data Warehouse

The script `adwc_dataload.sql` contains all the load commands necessary to load your unloaded Amazon Redshift data straight from S3 into your Autonomous Data Warehouse.
Execute the script while connected to your Autonomous Data Warehouse 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
```
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 Data Warehouse**

1. **Drop schema SQLDEV_MIGREPOS**

   As part of the schema migration the Migration Assistant installs a minimal migration repository in the target Autonomous Data Warehouse. 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_DWS_CREDS** in your Autonomous Data Warehouse 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 Data Warehouse**

   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.
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.

Topics

• Star Schema Benchmark Queries
• Star Schema Benchmark Analytic Views

Star Schema Benchmark Queries

```sql
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_category 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_region = '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 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)
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;
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.

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 Autonomous Data Warehouse database, the collection metadata includes the following information (regardless of which SODA implementation you use to create the default collection):

```json
{
    "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"
    }
}
```
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:

```java
OracleDocument metadata = db.createDocumentFromString("metadata_string");
OracleCollection col = db.admin().createCollection("myCustomColl", metadata);
```

In this example, for the metadata string you can use the default metadata as the starting point, and customize the following settings:

- **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 shows how to request a custom media type column. The `mediaTypeColumn` name is specified with the value `YOUR_MEDIA_TYPE_COLUMN_NAME`. The `"jsonFormat" : "OSON"` is automatically added and you'll see it in the metadata for a collection you create (for example, if you call `getMetadata` to view your metadata).

```json
{
   "keyColumn" :
   {
      "name" : "ID",
      "sqlType" : "VARCHAR2",
      "maxLength" : 255,
      "assignmentMethod" : "UUID"
   },

   "mediaTypeColumn" :
```
{  
   "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}
Obtain Tenancy Details to File 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.

If you need to file a service request use Oracle Cloud Support or contact your support representative and provide the following information:

- Database name
- Region
- Tenant OCID
- Database OCID
- Compartment OCID

If you are connected to the database you can obtain these details by querying the CLOUD_IDENTITY column of the V$PDBS view.

For example:

```
SELECT cloud_identity FROM v$pdbs;
{
  "DATABASE_NAME" : "DBxxxxxxxxxxxxx",
  "REGION" : "us-phoenix-1",
  "TENANT_OCID" : "OCID1.TENANCY.REGION1..ID1",
  "DATABASE_OCID" : "OCID1.AUTONOMOUSDATABASE.OC1.SEA.ID2",
  "COMPARTMENT_OCID" : "ocid1.tenancy.region1..ID3"
}
```

**Note:**

If your Autonomous Database instance was created before this 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.

See Restart Autonomous Data Warehouse for information on using restart from the Oracle Cloud Infrastructure console.

See RestartAutonomousDatabase for information on the restart API.

See Resource Identifiers for information on Oracle Cloud Identifiers.