Oracle® Cloud

Using Oracle Autonomous Database on Shared Exadata Infrastructure
Contents

Preface

Audience xxvii
Documentation Accessibility xxvii
Related Documents xxvii
Conventions xxviii

Part I Using Autonomous Database

1 Getting Started with Autonomous Database

Before You Begin with Oracle Autonomous Database 1-1
What is Oracle Autonomous Database? 1-2
Key Features of Autonomous Database 1-3
About Autonomous Database Workload Types 1-5
  About Autonomous Database for Analytics and Data Warehousing 1-5
  About Autonomous Database for Transaction Processing and Mixed Workloads 1-6
  About Autonomous JSON Database 1-7
  About Oracle APEX Application Development 1-8
Autonomous Database Region Availability 1-8
Always Free Autonomous Database 1-8
Security and Authentication in Oracle Autonomous Database 1-11
  Configuration Management 1-12
  Data Encryption 1-12
  Data Access Control 1-13
    Client Access Control 1-13
    Database User Access Control 1-13
    Oracle Cloud User Access Control 1-14
Auditing 1-15
  Auditing Service Level Activities 1-16
  Auditing Database Activities 1-16
Assessing the Security of Your Database and its Data 1-16
Regulatory Compliance Certification 1-17
2 Connecting to Autonomous Database

About Connecting to an Autonomous Database Instance 2-1
  Secure Connections to Autonomous Database 2-2
    About Mutual TLS (mTLS) Authentication 2-2
    About TLS Authentication 2-3
  Connecting to Autonomous Database Through a Firewall 2-4
  Using Application Continuity 2-5

Connect to Autonomous Database Using a Client Application 2-5
  About Connecting to Autonomous Database Using a Client Application 2-5
  Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections with Wallets (mTLS) 2-6
  Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections Using TLS Authentication 2-9
  Prepare for JDBC Thin Connections 2-10
    Set JVM Networking Properties 2-11
  Connect Microsoft .NET, Visual Studio Code, and Visual Studio with a Wallet (mTLS) 2-11
  Connect Microsoft .NET, Visual Studio Code, and Visual Studio without a Wallet 2-12
  Connect Python, Node.js, and other Scripting Languages (mTLS) 2-13
  Connect Python, Node.js, and other Scripting Languages without a Wallet 2-17

Download Database Connection Information 2-19
  Download Client Credentials (Wallets) 2-19
  Wallet README File 2-22
  View TNS Names and Connection Strings for an Autonomous Database Instance 2-23

Connect to Autonomous Database Using Oracle Database Tools 2-26
  Connect Oracle SQL Developer with a Wallet (mTLS) 2-26
Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint
Database Link Notes
Create Database Links to Non-Oracle Databases
Create Database Links to Non-Oracle Databases with Oracle-Managed Heterogeneous Connectivity
Create Database Links to Non-Oracle Databases with Customer-Managed Heterogeneous Connectivity
Create Database Links from Other Databases to Autonomous Database
Drop Database Links

3 Loading Data with Autonomous Database

About Data Loading
Load Data with Oracle Database Actions
Load Data from Files in the Cloud
Create Credentials and Copy Data into an Existing Table
Create Credentials and Load Data Pump Dump Files into an Existing Table
Load JSON on Autonomous Database
About Loading JSON Documents
Load a JSON File of Line-Delimited Documents into a Collection
Load an Array of JSON Documents into a Collection
Monitor and Troubleshoot COPY_COLECTION Loads
Objects That Extend JSON Scalars
Create Credentials and Copy JSON Data into an Existing Table
Monitor and Troubleshoot Loads
List Credentials
Delete Credentials
Example Loading Data from a Fixed Width File
Load a Fixed-Width File into a New Table
Import Data Using Oracle Data Pump on Autonomous Database
Export Your Existing Oracle Database to Import into Autonomous Database
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
Load Data from Local Files with Oracle Database Actions
Load Data into Existing Autonomous Database Table with Oracle Database Actions
Use Oracle GoldenGate to Replicate Data to Autonomous Database
Load Data from Local Files Using SQL*Loader
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Querying External Data with Autonomous Database</td>
<td></td>
</tr>
<tr>
<td>Query External Data</td>
<td>4-2</td>
</tr>
<tr>
<td>External Table Metadata Columns</td>
<td>4-4</td>
</tr>
<tr>
<td>Query External Data with ORC, Parquet, or Avro Source Files</td>
<td>4-4</td>
</tr>
<tr>
<td>Query External Tables with Partitioning Specified in Source Files</td>
<td>4-7</td>
</tr>
<tr>
<td>About External Tables with Source File Partitioning</td>
<td>4-8</td>
</tr>
<tr>
<td>External Partitioning: CSV Source Files with Hive-style Folders</td>
<td>4-11</td>
</tr>
<tr>
<td>External Partitioning: CSV Source Files with Simple Folders</td>
<td>4-11</td>
</tr>
<tr>
<td>External Partitioning: Parquet Source Files with Hive-style Folders</td>
<td>4-12</td>
</tr>
<tr>
<td>External Partitioning: Parquet with Simple Folders</td>
<td>4-13</td>
</tr>
<tr>
<td>Query External Partitioned Data with Hive Format Source File Organization</td>
<td>4-14</td>
</tr>
<tr>
<td>Query External Partitioned Data with Folder Format Source File Organization</td>
<td>4-17</td>
</tr>
<tr>
<td>Refresh External Partitioned Tables with Updated or Deleted Source Files</td>
<td>4-20</td>
</tr>
<tr>
<td>Query External Partitioned Data (with Partitioning Clause)</td>
<td>4-20</td>
</tr>
<tr>
<td>Query Hybrid Partitioned Data</td>
<td>4-22</td>
</tr>
<tr>
<td>Query External Data Pump Dump Files</td>
<td>4-24</td>
</tr>
<tr>
<td>Query Big Data Service Hadoop (HDFS) Data from Autonomous Database</td>
<td>4-26</td>
</tr>
<tr>
<td>Query External Data with Data Catalog</td>
<td>4-26</td>
</tr>
<tr>
<td>About Querying with Data Catalog</td>
<td>4-27</td>
</tr>
<tr>
<td>Concepts Related to Querying with Data Catalog</td>
<td>4-27</td>
</tr>
<tr>
<td>Synchronization Mapping</td>
<td>4-29</td>
</tr>
<tr>
<td>Typical Workflow with Data Catalog</td>
<td>4-31</td>
</tr>
<tr>
<td>Example: MovieStream Scenario</td>
<td>4-34</td>
</tr>
<tr>
<td>Example: Partitioned Data Scenario</td>
<td>4-40</td>
</tr>
<tr>
<td>Validate External Data</td>
<td>4-47</td>
</tr>
<tr>
<td>Validate External Partitioned Data</td>
<td>4-49</td>
</tr>
<tr>
<td>Validate Hybrid Partitioned Data</td>
<td>4-50</td>
</tr>
<tr>
<td>View Logs for Data Validation</td>
<td>4-51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 Creating Dashboards, Reports, and Notebooks with Autonomous Database</td>
<td></td>
</tr>
<tr>
<td>Create Dashboards and Reports to Analyze and Visualize Your Data</td>
<td>5-1</td>
</tr>
<tr>
<td>Use Oracle Analytics Desktop with Autonomous Database</td>
<td>5-1</td>
</tr>
<tr>
<td>Use Oracle Analytics Cloud with Autonomous Database</td>
<td>5-2</td>
</tr>
<tr>
<td>Create Notebooks, Workspaces, and Projects with Oracle Machine Learning Notebooks</td>
<td>5-3</td>
</tr>
<tr>
<td>Using Oracle Machine Learning Notebooks on Autonomous Database</td>
<td>5-3</td>
</tr>
<tr>
<td>Work with Oracle Machine Learning Notebooks for Data Access, Analysis, and Discovery</td>
<td>5-3</td>
</tr>
</tbody>
</table>
## Exporting Data from Autonomous Database to Object Store or to Other Oracle Databases

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move Data with Data Pump Export to an Autonomous Database Directory</td>
<td>6-1</td>
</tr>
<tr>
<td>Use Data Pump to Create a Dump File Set on Autonomous Database</td>
<td>6-2</td>
</tr>
<tr>
<td>Move Dump File Set from Autonomous Database to Your Cloud Object Store</td>
<td>6-4</td>
</tr>
<tr>
<td>Move Data with Data Pump Export to Object Store</td>
<td>6-5</td>
</tr>
<tr>
<td>Use Oracle Data Pump to Export Data to Object Store Using CREDENTIAL Parameter (Version 19.9 or Later)</td>
<td>6-6</td>
</tr>
<tr>
<td>Use Oracle Data Pump to Export Data to Object Store Setting DEFAULT_CREDENTIAL Property</td>
<td>6-9</td>
</tr>
<tr>
<td>Move Data to Object Store as Oracle Data Pump Files Using EXPORT_DATA</td>
<td>6-12</td>
</tr>
<tr>
<td>Download Dump Files, Run Data Pump Import, and Clean Up Object Store</td>
<td>6-14</td>
</tr>
<tr>
<td>Access Log Files for Data Pump Export</td>
<td>6-15</td>
</tr>
<tr>
<td>Move Data to Object Store as CSV, JSON, or XML Using EXPORT_DATA</td>
<td>6-15</td>
</tr>
<tr>
<td>Export JSON Data to Cloud Object Storage</td>
<td>6-16</td>
</tr>
<tr>
<td>Export Data as CSV to Cloud Object Storage</td>
<td>6-17</td>
</tr>
<tr>
<td>Export Data as XML to Cloud Object Storage</td>
<td>6-19</td>
</tr>
<tr>
<td>File Naming with Text Output (CSV, JSON, or XML)</td>
<td>6-20</td>
</tr>
<tr>
<td>Oracle GoldenGate Capture for Oracle Autonomous Database</td>
<td>6-23</td>
</tr>
</tbody>
</table>

## Developing RESTful Services in Autonomous Database

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>About Oracle REST Data Services in Autonomous Database</td>
<td>7-1</td>
</tr>
<tr>
<td>Access RESTful Services and SODA for REST</td>
<td>7-2</td>
</tr>
<tr>
<td>Develop with Oracle REST Data Services on Autonomous Database</td>
<td>7-2</td>
</tr>
<tr>
<td>Use SODA for REST with Autonomous Database</td>
<td>7-3</td>
</tr>
<tr>
<td>Overview of Using SODA for REST</td>
<td>7-3</td>
</tr>
<tr>
<td>Load Purchase-Order Sample Data Using SODA for REST</td>
<td>7-5</td>
</tr>
<tr>
<td>Use SODA for REST with OAuth Client Credentials</td>
<td>7-6</td>
</tr>
<tr>
<td>Access Oracle REST Data Services, Oracle APEX, and Developer Tools Using a Vanity URL</td>
<td>7-9</td>
</tr>
<tr>
<td>About Customer Managed Oracle REST Data Services on Autonomous Database</td>
<td>7-9</td>
</tr>
</tbody>
</table>

## Using Oracle Database API for MongoDB

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure Access for MongoDB</td>
<td>8-1</td>
</tr>
<tr>
<td>User Management for MongoDB</td>
<td>8-4</td>
</tr>
<tr>
<td>Create a Test Autonomous Database User for MongoDB</td>
<td>8-5</td>
</tr>
<tr>
<td>Connect MongoDB Applications to Autonomous Database</td>
<td>8-6</td>
</tr>
<tr>
<td>Retrieve the Autonomous Database Connection String</td>
<td>8-6</td>
</tr>
<tr>
<td>Test Connection Using the Command Line</td>
<td>8-7</td>
</tr>
</tbody>
</table>
9 Creating Applications with Oracle APEX in Autonomous Database

- About Oracle APEX 9-1
- Access Oracle APEX Administration Services 9-2
- Create Oracle APEX Workspaces in Autonomous Database 9-4
- Access Oracle APEX App Builder 9-5
- Create Oracle APEX Developer Accounts 9-5
- Load Data from the Cloud into Oracle APEX 9-6
- Use JSON Data with Oracle APEX 9-7
  - Create a View from JSON Data Guide 9-7
  - Create a View with JSON_TABLE Function 9-8
- Use Web Services with Oracle APEX 9-8
- Send Email from Oracle APEX 9-10
- Control Oracle APEX Upgrades 9-12
  - Defer Oracle APEX Upgrades 9-12
  - Apply Oracle APEX Upgrades 9-13
- Access Oracle APEX, Oracle REST Data Services, and Developer Tools Using a Vanity URL 9-15
- Restrictions and Limitations for Oracle APEX with Autonomous Database 9-15

10 Creating and Managing Directories on Autonomous Database

- Create Directory in Autonomous Database 10-1
- Drop Directory in Autonomous Database 10-2
- List Contents of Directory in Autonomous Database 10-3
- Copy Files Between Object Store and a Directory in Autonomous Database 10-3

11 Sending Email on Autonomous Database

- Send Email with Email Delivery Service on Autonomous Database 11-1
  - SMTP Send Email Sample Code 11-2
- Send Email with an Email Provider on a Private Endpoint 11-3

12 Oracle Extensions for IDEs

- Use Oracle Cloud Infrastructure Toolkit for Eclipse 12-1
- Use Oracle Developer Tools for Visual Studio 12-2
- Use Oracle Developer Tools for VS Code 12-3
13  **Using JSON Documents with Autonomous Database**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work with Simple Oracle Document Access (SODA) in Autonomous Database</td>
<td>13-1</td>
</tr>
<tr>
<td>SODA Notes</td>
<td>13-3</td>
</tr>
<tr>
<td>Work with JSON Documents Using SQL and PL/SQL APIs on Autonomous Database</td>
<td>13-3</td>
</tr>
<tr>
<td>Load JSON Documents with Autonomous Database</td>
<td>13-4</td>
</tr>
<tr>
<td>Import SODA Collection Data Using Oracle Data Pump Version 19.6 or Later</td>
<td>13-4</td>
</tr>
</tbody>
</table>

14  **Using Oracle Graph with Autonomous Database**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>About Oracle Graph Studio on Autonomous Database</td>
<td>14-1</td>
</tr>
<tr>
<td>Building Applications Using Graph APIs with Autonomous Database</td>
<td>14-1</td>
</tr>
</tbody>
</table>

15  **Using Oracle Spatial with Autonomous Database**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>About Oracle Spatial with Autonomous Database</td>
<td>15-1</td>
</tr>
<tr>
<td>Oracle Spatial Limitations with Autonomous Database</td>
<td>15-1</td>
</tr>
</tbody>
</table>

16  **Accessing Cloud Resources by Configuring Policies and Roles**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Resource Principal to Access Oracle Cloud Infrastructure Resources</td>
<td>16-1</td>
</tr>
<tr>
<td>About Using Resource Principal to Access Oracle Cloud Infrastructure Resources</td>
<td>16-1</td>
</tr>
<tr>
<td>Perform Prerequisites to Use Resource Principal with Autonomous Database</td>
<td>16-2</td>
</tr>
<tr>
<td>Enable Resource Principal to Access Oracle Cloud Infrastructure Resources</td>
<td>16-4</td>
</tr>
<tr>
<td>Disable Resource Principal on Autonomous Database</td>
<td>16-5</td>
</tr>
<tr>
<td>Use Resource Principal with DBMS_CLOUD</td>
<td>16-6</td>
</tr>
<tr>
<td>Use Amazon Resource Names (ARNs) to Access AWS Resources</td>
<td>16-7</td>
</tr>
<tr>
<td>About Using Amazon Resource Names (ARNs) to Access AWS Resources</td>
<td>16-7</td>
</tr>
<tr>
<td>About Steps to Configure ARN Usage with Autonomous Database</td>
<td>16-7</td>
</tr>
<tr>
<td>About Steps to Use ARNs with DBMS_CLOUD</td>
<td>16-9</td>
</tr>
<tr>
<td>Perform Autonomous Database Prerequisites to Use Amazon ARNs</td>
<td>16-9</td>
</tr>
<tr>
<td>Perform AWS Management Prerequisites to Use Amazon Resource Names (ARNs)</td>
<td>16-10</td>
</tr>
<tr>
<td>Create Credentials with ARN Parameters to Access AWS Resources</td>
<td>16-14</td>
</tr>
<tr>
<td>Update Credentials with ARN Parameters for AWS Resources</td>
<td>16-14</td>
</tr>
<tr>
<td>Use Azure Service Principal to Access Azure Resources</td>
<td>16-15</td>
</tr>
<tr>
<td>Enable Azure Service Principal</td>
<td>16-16</td>
</tr>
<tr>
<td>Provide Azure Application Consent and Assign Roles</td>
<td>16-17</td>
</tr>
<tr>
<td>Use Azure Service Principal with DBMS_CLOUD</td>
<td>16-19</td>
</tr>
<tr>
<td>Disable Azure Service Principal</td>
<td>16-20</td>
</tr>
<tr>
<td>Notes for Azure Service Principal</td>
<td>16-20</td>
</tr>
</tbody>
</table>
## Initialize a Cloud Code Repository

- Create and Manage a Cloud Code Repository
- Use File Operations with a Cloud Code Repository
- Use SQL Install Operations with a Cloud Code Repository

### Part II  Load, Explore, and Analyze Data with Data Tools on Autonomous Database

#### 22  The Data Load Page

- Checking Data Load Jobs
- Exploring Data
- Managing Cloud Storage Connections
- Loading Data with Autonomous Database
  - Loading Data From Local Files
  - Loading Data from Other Databases
  - Loading Data from Cloud Storage
    - Manage Cloud Storage Links for Data Load Jobs
    - Prepare the Data Load Job
    - Add Files or Folders for the Data Load Job
    - Enter Details for the Data Load Job
    - Run the Data Load Job
    - View Details About the Data Load Job After It Is Run
    - View the Table Resulting from the Data Load Job
- Linking Data
  - Linking to Other Databases
  - Linking to Objects in Cloud Storage
    - Manage Cloud Storage Links for Data Link Jobs
    - Prepare the Data Link Job
    - Add Files or Folders for the Data Link Job
    - Enter Details for the Data Link Job
    - Run the Data Link Job
    - View Details About the Data Link Job After It Is Run
    - View the Table Resulting from the Data Link Job
- Feeding Data
  - Creating a Notification-Based Live Table Feed
  - Creating a Notification-Based Live Table Feed using Amazon Simple Storage Service (S3)
  - Creating a Notification-Based Live Table Feed using Microsoft Azure
### The Catalog Page

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>About the Catalog Page</td>
<td>23-1</td>
</tr>
<tr>
<td>Sorting and Filtering Entities</td>
<td>23-5</td>
</tr>
<tr>
<td>Searching for Entities</td>
<td>23-6</td>
</tr>
<tr>
<td>Viewing Entity Details</td>
<td>23-14</td>
</tr>
</tbody>
</table>

### The Data Insights Page

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>About Insights</td>
<td>24-1</td>
</tr>
<tr>
<td>Generate Insights and View Reports</td>
<td>24-2</td>
</tr>
</tbody>
</table>

### The Data Analysis Tool

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Searching and obtaining information about Analytic Views</td>
<td>25-2</td>
</tr>
<tr>
<td>Creating Analytic Views</td>
<td>25-3</td>
</tr>
<tr>
<td>Using Calculation Templates</td>
<td>25-14</td>
</tr>
<tr>
<td>Cumulative Aggregates</td>
<td>25-14</td>
</tr>
<tr>
<td>Prior and Future Period</td>
<td>25-15</td>
</tr>
<tr>
<td>Period to Date</td>
<td>25-16</td>
</tr>
<tr>
<td>Parallel Period</td>
<td>25-17</td>
</tr>
<tr>
<td>Moving Aggregates</td>
<td>25-17</td>
</tr>
<tr>
<td>Share</td>
<td>25-18</td>
</tr>
<tr>
<td>Rank</td>
<td>25-19</td>
</tr>
<tr>
<td>Analyzing data</td>
<td>25-20</td>
</tr>
</tbody>
</table>

### Part III  Managing and Monitoring Autonomous Database

### Controlling Your Autonomous Database Instance

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provision Autonomous Database</td>
<td>26-1</td>
</tr>
<tr>
<td>Start Autonomous Database</td>
<td>26-5</td>
</tr>
<tr>
<td>Stop Autonomous Database</td>
<td>26-5</td>
</tr>
<tr>
<td>Restart Autonomous Database</td>
<td>26-6</td>
</tr>
<tr>
<td>Rename Autonomous Database</td>
<td>26-7</td>
</tr>
<tr>
<td>Update the Display Name for an Autonomous Database Instance</td>
<td>26-9</td>
</tr>
<tr>
<td>Terminate an Autonomous Database Instance</td>
<td>26-10</td>
</tr>
<tr>
<td>Add CPU or Storage Resources or Enable Auto Scaling</td>
<td>26-10</td>
</tr>
<tr>
<td>Remove CPU or Storage Resources or Disable Auto Scaling</td>
<td>26-12</td>
</tr>
<tr>
<td>Use Auto Scaling</td>
<td>26-14</td>
</tr>
<tr>
<td>OCPU Auto Scaling</td>
<td>26-15</td>
</tr>
</tbody>
</table>
27  Managing Users on Autonomous Database

Create Users on Autonomous Database 27-1
- Create Users on Autonomous Database with Database Actions 27-1
- Create Users on Autonomous Database - Connecting with a Client Tool 27-3
Unlock User Accounts on Autonomous Database 27-4
About User Passwords on Autonomous Database 27-4
Manage the Administrator Account on Autonomous Database 27-5
Set the ADMIN Password in Autonomous Database 27-5
Unlock the ADMIN Account in Autonomous Database 27-6
Manage User Roles and Privileges on Autonomous Database 27-6
- Manage Users and User Roles on Autonomous Database - Connecting with Database Actions 27-6
- Manage User Privileges on Autonomous Database - Connecting with a Client Tool 27-8
Remove Users on Autonomous Database 27-9
Manage User Profiles with Autonomous Database 27-10
- Manage Password Complexity on Autonomous Database 27-11
- Gradual Database Password Rollover for Applications 27-13
Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database 27-13
- Create User 27-14
- Add Existing Database User Account to Oracle Machine Learning Components 27-15
Use Identity and Access Management (IAM) Authentication with Autonomous Database 27-16
- About Identity and Access Management (IAM) Authentication with Autonomous Database 27-16
- IAM Database Password Authentication 27-17
- Identity and Access Management (IAM) SSO Token Based Authentication 27-17
- Prerequisites for Identity and Access Management (IAM) Authentication on Autonomous Database 27-18
Enable Identity and Access Management (IAM) Authentication on Autonomous Database 27-18
Create Identity and Access Management (IAM) Groups and Policies for IAM Users 27-20
Add IAM Users on Autonomous Database 27-21
Add IAM Roles on Autonomous Database 27-22
Create IAM Database Password for IAM Users 27-23
Connect to Autonomous Database with Identity and Access Management (IAM) Authentication 27-23
   About Connecting to an Oracle Autonomous Database Instance Using IAM 27-24
   Configuring a Client Connection for SQL*Plus That Uses an IAM Database Password 27-26
   Configuring a Client Connection for SQL*Plus That Uses an IAM Token 27-26
   Use Instance Principal to Access Autonomous Database with Identity and Access Management (IAM) Authentication 27-28
Configure Proxy Authentication 27-28
   About Configuring Proxy Authentication 27-28
   Configure Proxy Authentication for the IAM User 27-29
   Validate the IAM User Proxy Authentication 27-29
Disable Identity and Access Management (IAM) Authentication on Autonomous Database 27-30
Notes for Using Autonomous Database Tools with Identity and Access Management (IAM) Authentication 27-31
Use Azure Active Directory (Azure AD) with Autonomous Database 27-31
   About Authorizing Microsoft Azure AD Users for an Oracle Autonomous Database 27-31
      Architecture of the Microsoft Azure AD Integration with an Oracle Autonomous Database 27-32
      Azure AD Users Mapping to the Oracle Autonomous Database 27-33
      Use Cases for Connecting to an Oracle Autonomous Database Using Azure AD 27-35
Enable Azure AD Authentication on Autonomous Database 27-35
   Registering the Oracle Autonomous Database Instance with a Microsoft Azure AD Tenancy 27-35
   Managing App Roles in Microsoft Azure AD 27-39
   Configuring Azure AD as an External Identity Provider for Autonomous Database 27-41
Role and Schema Mapping for Azure AD Authentication on Autonomous Database 27-43
   Exclusively Mapping an Oracle Database Schema to a Microsoft Azure AD User 27-43
   Mapping a Shared Oracle Schema to an App Role 27-43
   Mapping an Oracle Database Global Role to an App Role 27-44
   Azure AD Client Configuration and Access for Autonomous Database 27-44
Use Microsoft Active Directory with Autonomous Database 27-44
   Configure CMU with Microsoft Active Directory on Autonomous Database 27-45
   Add Microsoft Active Directory Roles on Autonomous Database 27-47
   Add Microsoft Active Directory Users on Autonomous Database 27-48
   Connect to Autonomous Database with Active Directory User Credentials 27-50
Tools Restrictions with Active Directory on Autonomous Database 27-51
Verify Active Directory User Connection Information with Autonomous Database 27-51
Remove Active Directory Users and Roles on Autonomous Database 27-52
Disable Active Directory Access on Autonomous Database 27-52
Use Kerberos Authentication with Autonomous Database 27-53
About Kerberos Authentication 27-53
Components of the Kerberos Authentication System 27-53
Enable Kerberos Authentication on Autonomous Database 27-54
Disable Kerberos Authentication on Autonomous Database 27-55
Notes for Kerberos Authentication on Autonomous Database 27-55

28 Managing and Monitoring Performance of Autonomous Database

Monitor the Performance of Autonomous Database 28-1
Use Database Actions to Monitor Activity and Utilization 28-2
Database Monitor Overview 28-2
Database Monitor Activity 28-7
Use Database Actions to Monitor Active Session History Analytics and SQL Statements 28-8
The Performance Hub Page 28-8
Manage Concurrency and Priorities on Autonomous Database 28-10
Database Service Names for Autonomous Data Warehouse 28-10
Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database 28-11
Idle Time Limits 28-12
Service Concurrency 28-13
Service Concurrency Limits for Data Warehouse Workloads 28-13
Service Concurrency Limits for Transaction Processing and JSON Database Workloads 28-14
Change MEDIUM Service Concurrency Limit 28-14
Change MEDIUM Service Concurrency Limit with PL/SQL Procedure UPDATE_PLAN_DIRECTIVE 28-16
Change MEDIUM Service Concurrency Limit Notes 28-17
Predefined Job Classes with Oracle Scheduler 28-17
Manage CPU/IO Shares on Autonomous Database 28-18
Manage Runaway SQL Statements on Autonomous Database 28-21
Manage Optimizer Statistics on Autonomous Database 28-22
Manage Optimizer Statistics with Data Warehouse Workloads 28-22
Manage Optimizer Statistics with Transaction Processing and JSON Database Workloads 28-23
Manage Automatic Indexing on Autonomous Database 28-24
Manage Automatic Partitioning on Autonomous Database 28-24
About Automatic Partitioning 28-25
### How Automatic Partitioning Works
- Configure Automatic Partitioning 28-26
- Use Automatic Partitioning 28-26
- Generate Automatic Partitioning Reports 28-28
- Example Automatic Partitioning Scenarios 28-29
- Data Dictionary Views for Automatic Partitioning 28-30
  - DBMS_AUTO_PARTITION DBA_AUTO_PARTITION_CONFIG View 28-30
  - DBMS_AUTO_PARTITION DBA_AUTO_PARTITION_RECOMMENDATIONS View 28-31
  - DBMS_AUTO_PARTITION Updates to Existing Views 28-32
- Monitor Autonomous Database with Performance Hub 28-32
  - Time Selector 28-33
  - AWR Report 28-34
  - ASH Analytics 28-34
  - SQL Monitoring 28-35
  - ADDM 28-35
  - Workload 28-36
  - Blocking Sessions 28-36
- Monitor the Performance of Autonomous Database with Oracle Management Cloud 28-37
- Use Database Management Service to Monitor Databases 28-37
- Monitor Performance with Autonomous Database Metrics 28-38
  - View Metrics for an Autonomous Database Instance 28-38
  - View Metrics for Autonomous Databases in a Compartment 28-40
- Autonomous Database Metrics and Dimensions 28-41
- Perform SQL Tracing on Autonomous Database 28-42
  - Configure SQL Tracing on Autonomous Database 28-42
  - Enable SQL Tracing on Autonomous Database 28-44
  - Disable SQL Tracing on Autonomous Database 28-46
  - View Trace File Saved to Cloud Object Store on Autonomous Database 28-46
  - View Trace Data in SESSION_CLOUD_TRACE View on Autonomous Database 28-48
- Monitor Autonomous Database Availability 28-49
- Monitor Regional Availability of Autonomous Databases 28-50
- Service Console Replacement with Database Actions 28-52
- Available Metrics: oci_autonomous_database 28-53

### 29 Backing Up and Restoring Autonomous Database

- About Backup and Recovery on Autonomous Database 29-1
- Restore and Recover your Autonomous Database 29-2
- Manual Backups on Autonomous Database 29-5
  - Configure Manual Backups on Autonomous Database 29-5
  - Perform Manual Backups on Autonomous Database 29-7
# Cloning and Moving an Autonomous Database

Clone an Autonomous Database Instance 30-1
Clone Autonomous Database from a Backup 30-5
Notes for Cloning Autonomous Database 30-10
  - Resource Management Rules and Performance Data for a Cloned Database 30-11
  - Optimizer Statistics for a Cloned Database 30-11
Move an Autonomous Database to a Different Compartment 30-12

# Wallet Security, Using Oracle Data Safe, Oracle Database Vault, and Changing License Type

Rotate Wallets for Autonomous Database 31-1
  - About Wallet Rotation 31-1
  - Rotate Wallets with Immediate Rotation 31-2
  - Rotate Wallets with Grace Period 31-3
View and Update Your License and Oracle Database Edition on Autonomous Database 31-6
Update Always Free Instance to Paid with Autonomous Database 31-8
Safeguard Your Data with Data Safe on Autonomous Database 31-8
Use Oracle Database Vault with Autonomous Database 31-8
  - Oracle Database Vault Users and Roles on Autonomous Database 31-9
  - Enable Oracle Database Vault on Autonomous Database 31-9
  - Disable Oracle Database Vault on Autonomous Database 31-10
  - Disable User Management with Oracle Database Vault on Autonomous Database 31-11
  - Enable User Management with Oracle Database Vault on Autonomous Database 31-11

# Managing Encryption Keys on Autonomous Database

About Master Encryption Key Management on Autonomous Database 32-1
  - Oracle-Managed Master Encryption Keys on Autonomous Database 32-2
  - Customer-Managed Encryption Keys on Autonomous Database 32-2
  - About Customer-Managed Encryption Key Rotation on Autonomous Database 32-3
Prerequisites to Use Customer-Managed Encryption Keys on Autonomous Database 32-3
Use Customer-Managed Encryption Keys on Autonomous Database 32-6
  - Switch to Oracle-Managed Encryption Keys on Autonomous Database 32-7
  - View History for Customer-Managed Encryption Keys on Autonomous Database 32-8
Notes for Using Customer-Managed Keys with Autonomous Database 32-9
  - Caution for Customer-Managed Encryption Keys when Oracle Cloud Infrastructure Vault is Unreachable 32-9
## 33 Configuring Network Access with Access Control Rules (ACLs) and Private Endpoints

- Configuring Network Access with Access Control Rules (ACLs)
  - Configure Access Control Lists When You Provision or Clone an Instance
  - Configure Access Control Lists for an Existing Autonomous Database Instance
  - Change from Private to Public Endpoints with Autonomous Database
  - Access Control List Restrictions and Notes
- Configuring Network Access with Private Endpoints
  - Configure Private Endpoints
    - Prerequisite: IAM Policies Required to Manage Private Endpoints
  - Change from Public to Private Endpoints with Autonomous Database
- Enhanced Security for Outbound Connections with Private Endpoints
- Private Endpoints Notes
- Private Endpoints Configuration Examples on Autonomous Database
  - Example: Connecting from Inside Oracle Cloud Infrastructure VCN
  - Example: Connecting from Your Data Center to Autonomous Database
- Update Network Options to Allow TLS or Require Only Mutual TLS (mTLS) Authentication on Autonomous Database

## 34 Managing and Viewing Maintenance Windows, Patching Options, Time Zone Data Updates, Work Requests, Events and Notifications, and Customer Contacts

- About Work Requests
- View Patch and Maintenance Window Information, Set the Patch Level
  - About Scheduled Maintenance and Patching
  - View Maintenance Event History
  - View Patch Details
  - Set the Patch Level
Using Standby Databases with Autonomous Database for Disaster Recovery

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>About Standby Databases</td>
<td>35-1</td>
</tr>
<tr>
<td>Autonomous Data Guard with Local Standby</td>
<td>35-2</td>
</tr>
<tr>
<td>Autonomous Data Guard with Cross-Region Standby</td>
<td>35-3</td>
</tr>
<tr>
<td>Autonomous Data Guard Database Region and Role</td>
<td>35-5</td>
</tr>
<tr>
<td>Autonomous Data Guard Failover and Switchover with Cross-Region Standby and No Local Standby</td>
<td>35-6</td>
</tr>
<tr>
<td>Autonomous Data Guard Failover and Switchover with Local Standby and Cross-Region Standby</td>
<td>35-6</td>
</tr>
<tr>
<td>Autonomous Data Guard Database Cross Region Backup and Restore</td>
<td>35-7</td>
</tr>
<tr>
<td>Autonomous Data Guard and Primary Region Wallets</td>
<td>35-7</td>
</tr>
<tr>
<td>Autonomous Data Guard Recovery Time Objective (RTO) and Recovery Point Objective (RPO)</td>
<td>35-8</td>
</tr>
<tr>
<td>Autonomous Data Guard Operations</td>
<td>35-8</td>
</tr>
<tr>
<td>Autonomous Database Standby Database State</td>
<td>35-9</td>
</tr>
<tr>
<td>Autonomous Data Guard Events with Autonomous Database</td>
<td>35-10</td>
</tr>
<tr>
<td>Enable a Standby Database</td>
<td>35-11</td>
</tr>
<tr>
<td>Add a Standby Database</td>
<td>35-13</td>
</tr>
<tr>
<td>Disable a Standby Database</td>
<td>35-15</td>
</tr>
<tr>
<td>Disable a Local Standby Database</td>
<td>35-15</td>
</tr>
<tr>
<td>Terminate a Cross-Region (Remote) Standby Database</td>
<td>35-17</td>
</tr>
<tr>
<td>Verify Autonomous Data Guard is Disabled</td>
<td>35-19</td>
</tr>
<tr>
<td>Perform a Switchover</td>
<td>35-20</td>
</tr>
<tr>
<td>Automatic Failover with a Standby Database</td>
<td>35-22</td>
</tr>
<tr>
<td>Manual Failover with a Standby Database</td>
<td>35-23</td>
</tr>
<tr>
<td>Events and Notifications for a Standby Database</td>
<td>35-25</td>
</tr>
<tr>
<td>Use the API</td>
<td>35-26</td>
</tr>
<tr>
<td>Autonomous Data Guard Notes</td>
<td>35-26</td>
</tr>
<tr>
<td>Autonomous Data Guard Paired Regions</td>
<td>35-28</td>
</tr>
</tbody>
</table>
### Using Refreshable Clones with Autonomous Database

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>About Refreshable Clones on Autonomous Database</td>
<td>36-1</td>
</tr>
<tr>
<td>Refreshable Clone Features</td>
<td>36-1</td>
</tr>
<tr>
<td>Refreshable Clone Operations</td>
<td>36-2</td>
</tr>
<tr>
<td>Refreshable Clone Lifecycle States</td>
<td>36-3</td>
</tr>
<tr>
<td>Refreshable Clone Refresh Timing and Disconnecting from the Source Database</td>
<td>36-4</td>
</tr>
<tr>
<td>Refreshable Clone Reconnect to the Source Database</td>
<td>36-4</td>
</tr>
<tr>
<td>Operations on an Autonomous Database with an Attached Refreshable Clone</td>
<td>36-5</td>
</tr>
<tr>
<td>Create a Refreshable Clone for an Autonomous Database Instance</td>
<td>36-5</td>
</tr>
<tr>
<td>View Refreshable Clones for an Autonomous Database Instance</td>
<td>36-9</td>
</tr>
<tr>
<td>Refresh a Refreshable Clone on Autonomous Database</td>
<td>36-10</td>
</tr>
<tr>
<td>Disconnect a Refreshable Clone from the Source Database</td>
<td>36-11</td>
</tr>
<tr>
<td>Reconnect a Refreshable Clone to the Source Database</td>
<td>36-13</td>
</tr>
<tr>
<td>Use the API</td>
<td>36-14</td>
</tr>
<tr>
<td>Refreshable Clone Notes</td>
<td>36-15</td>
</tr>
</tbody>
</table>

### Auditing Autonomous Database

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>About Auditing Autonomous Database</td>
<td>37-1</td>
</tr>
<tr>
<td>Audit Features on Autonomous Database</td>
<td>37-1</td>
</tr>
<tr>
<td>Audit Data on Autonomous Database</td>
<td>37-2</td>
</tr>
<tr>
<td>Default Audit Policies on Autonomous Database</td>
<td>37-3</td>
</tr>
<tr>
<td>Enable and Register Oracle Data Safe on Autonomous Database</td>
<td>37-3</td>
</tr>
<tr>
<td>Extend Audit Record Retention with Oracle Data Safe on Autonomous Database</td>
<td>37-4</td>
</tr>
<tr>
<td>View and Manage Oracle Data Safe Audit Trails on Autonomous Database</td>
<td>37-5</td>
</tr>
<tr>
<td>View and Manage Audit Policies with Oracle Data Safe on Autonomous Database</td>
<td>37-5</td>
</tr>
<tr>
<td>Generate Audit Reports with Data Safe on Autonomous Database</td>
<td>37-6</td>
</tr>
</tbody>
</table>

### Part IV Appendixes

#### A Autonomous Database Supplied Package Reference

<table>
<thead>
<tr>
<th>Package</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBMS_CLOUD Package</td>
<td>A-1</td>
</tr>
<tr>
<td>DBMS_CLOUD Subprograms and REST APIs</td>
<td>A-1</td>
</tr>
<tr>
<td>DBMS_CLOUD for Access Management</td>
<td>A-2</td>
</tr>
<tr>
<td>DBMS_CLOUD for Objects and Files</td>
<td>A-7</td>
</tr>
<tr>
<td>DBMS_CLOUD REST APIs</td>
<td>A-40</td>
</tr>
<tr>
<td>DBMS_CLOUD Package File URI Formats</td>
<td>A-53</td>
</tr>
<tr>
<td>Oracle Cloud Infrastructure Object Storage Native URI Format</td>
<td>A-53</td>
</tr>
<tr>
<td>Oracle Cloud Infrastructure Object Storage Swift URI Format</td>
<td>A-54</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>DBMS_CLOUD_REPO Package</td>
<td>A-104</td>
</tr>
<tr>
<td>DBMS_CLOUD_REPO Overview</td>
<td>A-105</td>
</tr>
<tr>
<td>DBMS_CLOUD_REPO Data Structures</td>
<td>A-106</td>
</tr>
<tr>
<td>DBMS_CLOUD_REPO Subprogram Groups</td>
<td>A-106</td>
</tr>
<tr>
<td>- DBMS_CLOUD_REPO Initialization Operations</td>
<td>A-106</td>
</tr>
<tr>
<td>- DBMS_CLOUD_REPO Repository Management Operations</td>
<td>A-106</td>
</tr>
<tr>
<td>- DBMS_CLOUD_REPO File Operations</td>
<td>A-107</td>
</tr>
<tr>
<td>- DBMS_CLOUD_REPO SQL Install Operations</td>
<td>A-107</td>
</tr>
<tr>
<td>Summary of DBMS_CLOUD_REPO Subprograms</td>
<td>A-107</td>
</tr>
<tr>
<td>CREATE_REPOSITORY Procedure</td>
<td>A-108</td>
</tr>
<tr>
<td>DELETE_FILE Procedure</td>
<td>A-108</td>
</tr>
<tr>
<td>DELETE_REPOSITORY Procedure</td>
<td>A-109</td>
</tr>
<tr>
<td>EXPORT_OBJECT Procedure</td>
<td>A-110</td>
</tr>
<tr>
<td>GET_FILE Procedure and Function</td>
<td>A-111</td>
</tr>
<tr>
<td>INIT_AWS_REPO Function</td>
<td>A-112</td>
</tr>
<tr>
<td>INIT_AZURE_REPO Function</td>
<td>A-112</td>
</tr>
<tr>
<td>INIT_GITHUB_REPO Function</td>
<td>A-113</td>
</tr>
<tr>
<td>INIT_REPO Function</td>
<td>A-114</td>
</tr>
<tr>
<td>INSTALL_FILE Procedure</td>
<td>A-114</td>
</tr>
<tr>
<td>INSTALL_SQL Procedure</td>
<td>A-115</td>
</tr>
<tr>
<td>LIST_FILES Function</td>
<td>A-116</td>
</tr>
<tr>
<td>LIST_REPOSITORIES Function</td>
<td>A-117</td>
</tr>
<tr>
<td>PUT_FILE Procedure</td>
<td>A-118</td>
</tr>
<tr>
<td>UPDATE_REPOSITORY Procedure</td>
<td>A-119</td>
</tr>
<tr>
<td>DBMS_DCAT Package</td>
<td>A-120</td>
</tr>
<tr>
<td>Managing the Data Catalog Connection</td>
<td>A-120</td>
</tr>
<tr>
<td>Data Catalog Users and Roles</td>
<td>A-120</td>
</tr>
<tr>
<td>Required Credentials and IAM Policies</td>
<td>A-121</td>
</tr>
<tr>
<td>SET_DATA_CATALOG_CREDENTIAL Procedure</td>
<td>A-123</td>
</tr>
<tr>
<td>SET_OBJECT_STORE_CREDENTIAL Procedure</td>
<td>A-124</td>
</tr>
<tr>
<td>SET_DATA_CATALOG_CONN Procedure</td>
<td>A-124</td>
</tr>
<tr>
<td>UNSET_DATA_CATALOG_CONN Procedure</td>
<td>A-125</td>
</tr>
<tr>
<td>Running Synchronizations</td>
<td>A-125</td>
</tr>
<tr>
<td>RUN_SYNC Procedure</td>
<td>A-126</td>
</tr>
<tr>
<td>CREATE_SYNC_JOB Procedure</td>
<td>A-130</td>
</tr>
<tr>
<td>DROP_SYNC_JOB Procedure</td>
<td>A-131</td>
</tr>
<tr>
<td>DROP_SYNCED_SCHEMAS Procedure</td>
<td>A-131</td>
</tr>
<tr>
<td>Data Catalog Views</td>
<td>A-131</td>
</tr>
<tr>
<td>- ALL_DCAT_GLOBAL_ACCESSIBLE_CATALOGS View</td>
<td>A-132</td>
</tr>
<tr>
<td>- ALL_DCAT_LOCAL_ACCESSIBLE_CATALOGS View</td>
<td>A-132</td>
</tr>
<tr>
<td>- ALL_DCAT_ASSETS View</td>
<td>A-133</td>
</tr>
</tbody>
</table>
Autonomous Database for Experienced Oracle Database Users

About Autonomous Database for Experienced Oracle Database Users B-1
Data Warehouse Workload with Autonomous Database B-1
Manage DML Performance and Compression for Data Warehouse Workloads B-2
Create Staging Tables for Data Warehouse Workloads B-3
Transaction Processing and JSON Database Workloads with Autonomous Database B-4
Autonomous Database Views B-5
Track Table and Partition Scan Access with Autonomous Database Views B-5
GV$TABLE_ACCESS_STATS and V$TABLE_ACCESS_STATS Views B-6
ALL_TABLE_ACCESS_STATS and DBA_TABLE_ACCESS_STATS Views B-6
USER_TABLE_ACCESS_STATS View B-6
Track Oracle Cloud Infrastructure Resources, Cost and Usage Reports with Autonomous Database Views B-7
Prerequisite Steps to Use OCI Resource Views B-7
OCI AUTONOMOUS_DATABASES View B-8
OCI_BUDGET_ALERT_RULES View B-11
OCI_BUDGET_SUMMARY View B-12
OCI_COST_DATA View B-13
OCI_OBJECTSTORAGE_BUCKETS View B-14
OCI_USAGE_DATA View B-15

Autonomous Database – Oracle Database Features B-16
Always Free Autonomous Database – Oracle Database 21c B-17
Always Free Autonomous Database Oracle Database 21c Features B-17
Always Free Autonomous Database Oracle Database 21c Notes B-22

Autonomous Database RMAN Recovery Catalog B-22
Use Autonomous Database as an RMAN Recovery Catalog B-22

Notes for Users Migrating from Other Oracle Databases B-23
  Initialization Parameters B-24
    SESSION_EXIT_ON_PACKAGE_STATE_ERROR B-25
    SYSDATE_AT_DBTIMEZONE B-25

SQL Commands B-27
Data Types B-31
PL/SQL Packages B-32
Oracle XML DB B-34
Oracle Text B-35
Oracle Flashback B-36
Oracle Database Real Application Security B-36
Oracle LogMiner B-37
Fast Ingest B-37
Choose a Character Set for Autonomous Database B-38
  Notes for Character Set Selection B-39
Database Features Unavailable in Autonomous Database B-40

C Migrating MySQL and Third-Party Databases to Autonomous Database

Migrating Amazon Redshift to Autonomous Database C-1
  Autonomous Database Redshift Migration Overview C-1
  Connect to Amazon Redshift C-2
  Connect to Autonomous Database 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-13
  Perform Post Migration Tasks C-15

Migrating MySQL to Autonomous Database C-15
Migrating Microsoft SQL Server or Sybase Adaptive Server to Autonomous Database C-16
Migrating IBM DB2 (UDB) Database to Autonomous Database C-16
D Sample Star Schema Benchmark (SSB) Queries and Analytic Views

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

E SODA Collection Metadata on Autonomous Database

- SODA Default Collection Metadata on Autonomous Database E-1
- SODA Customized Collection Metadata on Autonomous Database E-3

F Obtain Tenancy Details

- Database Name F-2
- Region F-2
- Tenancy OCID F-2
- Database OCID F-2
- Compartment OCID F-2
- Outbound IP Address F-2
Preface

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

Audience

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

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

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

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

Oracle Database 19c

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

Oracle Database 21c

For additional information, see these Oracle resources:

- Welcome to Oracle Cloud Infrastructure
- Oracle Cloud Infrastructure Object Storage
- Get Started Using Autonomous JSON Database
- GoldenGate Real-Time Data Replication in Cloud
- Using Oracle GoldenGate Cloud Service
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><strong>monospace</strong></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 Database

This part provides information on using Autonomous Database.

Topics

• Getting Started with Autonomous Database
• Connecting to Autonomous Database
• Loading Data with Autonomous Database
• Querying External Data with Autonomous Database
• Creating Dashboards, Reports, and Notebooks with Autonomous Database
• Exporting Data from Autonomous Database to Object Store or to Other Oracle Databases
• Developing RESTful Services in Autonomous Database
• Using Oracle Database API for MongoDB
• Creating Applications with Oracle APEX in Autonomous Database
• Creating and Managing Directories on Autonomous Database
• Sending Email on Autonomous Database
• Oracle Extensions for IDEs
• Using JSON Documents with Autonomous Database
• Using Oracle Graph with Autonomous Database
• Using Oracle Spatial with Autonomous Database
• Accessing Cloud Resources by Configuring Policies and Roles
• Using Application Continuity on Autonomous Database
• Calling Web Services from Autonomous Database
• Using Oracle Java on Autonomous Database
• Using Oracle Real Application Testing - Database Replay
• Using and Managing a Cloud Code Repository with Autonomous Database
Getting Started with Autonomous Database

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

Topics

• Before You Begin with Oracle Autonomous Database
• What is Oracle Autonomous Database?
• Key Features of Autonomous Database
• About Autonomous Database Workload Types
• Autonomous Database Region Availability
• Always Free Autonomous Database
• Security and Authentication in Oracle Autonomous Database
• Provision an Autonomous Database
• Use Sample Data Sets in Autonomous Database
• Build Reports and Dashboards with Analytics in Autonomous Database
• Use Oracle Machine Learning Notebooks with Autonomous Database
• Autonomous Database Tutorials
• Get Help, Search Forums, and Contact Support
• Availability Service Level Objectives (SLOs)

Before You Begin with Oracle Autonomous Database

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

Before you create an Autonomous Database:

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

  See Welcome to Oracle Cloud Infrastructure for more information.

• (Optional) if you want to leverage an object store for data loading you need your object store credentials to use with Oracle Autonomous Database, including a username and a password. For details on the required credentials, depending on the object store you want to use, see the following:

  – Oracle Cloud Infrastructure Object Storage, the username is your Oracle Cloud Infrastructure user name. The password is your auth token. See Working with Auth Tokens.
– **Oracle Cloud Infrastructure Object Storage Classic**, the *username* is your Oracle Cloud Infrastructure Classic user name and the *password* is your Oracle Cloud Infrastructure Classic password.

– **Amazon S3**, the *username* is your AWS access key ID and the *password* is your AWS secret access key. See [AWS Identity and Access Management](#).

– **Azure Blob Storage**, the *username* is your Azure storage account name and the *password* is an Azure storage account access key. See [About Azure storage accounts](#).

– **Amazon S3-Compatible**, such as Oracle Cloud Infrastructure Object Storage, Google Cloud Storage, and Wasabi Hot Cloud Storage. See [CREATE_CREDENTIAL Procedure](#) for more information.

– **GitHub Repository**, the *username* is your GitHub email and the *password* is your GitHub personal access token. See [Creating a personal access token](#) for more information.

• (Optional) Create a bucket for cloud storage manual backups:

If you want to manually back up Autonomous Database 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 Database](#).

### What is Oracle Autonomous Database?

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

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

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

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

Autonomous Database also includes the following:

• **Oracle APEX (APEX)**: a low-code development platform that enables you to build scalable, secure enterprise apps with world-class features.

• **Oracle REST Data Services (ORDS)**: a Java Enterprise Edition based data service that makes it easy to develop modern REST interfaces for relational data and JSON Document Store.
• Database Actions: is a web-based interface that uses Oracle REST Data Services to provide development, data tools, administration, and monitoring features for Autonomous Database.

• Oracle Machine Learning Notebooks: 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.

Key Features of Autonomous Database

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

Key Features

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

• **Fully elastic scaling**: Scale compute and storage independently to fit your database workload with no downtime:
  – Size the Autonomous Database to the exact compute and storage required
  – Scale the Autonomous Database on demand: Independently scale compute or storage
  – Shut off idle compute to save money

• **Auto scaling**: Allows your database to use more CPU and IO resources or to use additional storage automatically when the workload or storage demand requires additional resources:
  – Specify the number of OCPUs for Autonomous Database workloads.
  – OCPU auto scaling allows the database to use up to three times more CPU and IO resources, depending on workload requirements. OCPU auto scaling is enabled by default when you create an Autonomous Database.
  – Storage auto scaling allows the database to expand use up to three times the reserved base storage, depending on storage requirements. Storage auto scaling is disabled by default when you create an Autonomous Database.
  – You can manage scaling from the Oracle Cloud Infrastructure Console to enable or disable OCPU auto scaling or Storage auto scaling.

• **Autonomous Database supports**:
  – Existing applications, running in the cloud or on-premise
  – Connectivity via SQL*Net, JDBC, ODBC
  – Third-party data-integration tools
  – Oracle cloud services: Oracle Analytics Cloud, Oracle GoldenGate Marketplace, Oracle Integration Cloud Service, and others

• **High-performance queries and concurrent workloads**: Optimized query performance with preconfigured resource profiles for different types of users.
• **Oracle SQL**: Autonomous Database is compatible with existing applications that support Oracle Database.

• **Built-in web-based data analysis tool**: Web-based notebook tool for designing and sharing SQL based data-driven, interactive documents.

• **Database migration utility**: Easily migrate from MySQL, Amazon AWS Redshift, PostgreSQL, SQL Server, and other databases.

### Simple Cloud-based Data Loading

Autonomous Database provides:

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

### Oracle Database Actions

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

**• Development Tools**

– SQL Navigator and Worksheet: view objects and enter and run SQL and PL/SQL statements, and create database objects

– Data Modeler: provides an integrated version of Oracle SQL Developer Data Modeler with basic reporting features. You can create diagrams from existing schemas, retrieve data dictionary information, generate DDL statements, and export diagrams

– Develop and Secure RESTful Web Services

– Manage JSON Collections

**• Data Tools**

– Data Load: select data from your local computer, from tables in other databases, or from cloud storage and then add that data to new or existing tables or views

– Catalog: displays information about entities and dependencies in your data

– Data Insights: displays information about patterns and anomalies in your data

– Business Models: allows you to create a business model, edit a business model, and view information about a business model

**• Administration and Monitoring**

– Manage users

– Monitor databases

### Autonomous Data Guard

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

**SQL Developer Autonomous Database Support**

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

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

**Business Intelligence Tools Support**

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

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

### About Autonomous Database Workload Types

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

**Topics**

- About Autonomous Database for Analytics and Data Warehousing
- About Autonomous Database for Transaction Processing and Mixed Workloads
- About Autonomous JSON Database
- About Oracle APEX Application Development

### About Autonomous Database for Analytics and Data Warehousing

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

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

To get started you create an Autonomous Database with workload type Data Warehouse and specify the number of OCPUs and the storage capacity in TB's for the Autonomous Database.
You can use Autonomous Database with Oracle Analytics Cloud or Oracle Analytics Desktop to easily create visualizations and projects that reveal trends in your company's data and help you answer questions and discover important insights about your business.

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

About Autonomous Database for Transaction Processing and Mixed Workloads

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

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

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

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

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

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

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

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

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

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

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

No matter what kind of data your applications use, whether JSON or something else, you can take advantage of all Oracle Database features. This is true regardless of the kind of Oracle Autonomous Database you use.
JSON data is stored natively in the database. In a SODA collection on an Autonomous Database JSON data is stored in Oracle's native binary format, OSON.

About Oracle APEX Application Development

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

See Oracle APEX Application Development for more information.

Autonomous Database Region Availability

Commercial Regions and Availability Domains

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

Government Cloud

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

Always Free Autonomous Database

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

Sign Up with Oracle Cloud Free Tier

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

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

See Oracle Cloud Free Tier to start for free.

Resource Restrictions for Always Free Autonomous Database

- 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 Autonomous Database instances per Oracle Cloud Infrastructure tenancy. The Always Free Autonomous Database workload types are: Data Warehouse, Transaction Processing, JSON Database, and APEX Service. If you create 2 Always Free instances, they can be the same or different Autonomous Database workload types.
The HTTP interface for Always Free Autonomous Databases is rate limited to restrict the number of simultaneous service users. Approximately 3-6 simultaneous users can be supported across all of the APEX, Oracle REST Data Services, and Database Actions running on your Always Free Autonomous Databases. Additional simultaneous users beyond that may result in users encountering HTTP errors such as HTTP status code 429.

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

**Notes:**

- For details on Always Free Oracle APEX Application Development (APEX Service), see Always Free Oracle APEX Application Development.
- Always Free Autonomous Databases cannot be scaled manually or automatically beyond the fixed resource restrictions described above.
- The Maximum of 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 as a private endpoint and cannot reside within a Virtual Cloud Network (VCN). See Configure Private Endpoints When You Provision or Clone an Instance for further information on private endpoints.

**Oracle Database Version**

The available Database versions for Always Free Autonomous Database are: Oracle Database 19c or Oracle Database 21c.

**Regional Availability for Always Free Autonomous Database**

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

**Backup Functionality Not Available in Always Free Autonomous Database**

- Always Free Autonomous Databases do not support full database backups to your Oracle Cloud Infrastructure object storage.
- Always Free Autonomous Databases do not support restoring from full database backups.

See Backing Up and Restoring Autonomous Database for more information.
Autonomous Data Guard Not Available for Always Free Autonomous Database

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

Inactivity Monitoring and Database Stoppage

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

- After being inactive for 7 days, the database will be stopped automatically, preserving its stored data. Inactivity measurements leading up to 7 days are based on database connections. 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.

When you start an Always Free Autonomous Database from the stopped state, you need to wait about 5 minutes before attempting to connect to an APEX application or to an Oracle REST Data Services (ORDS) endpoint. If you attempt to connect before the background APEX and ORDS startup completes, then you may see HTTP error messages.

- On an Always Free database the Oracle Cloud Infrastructure console shows banner alerts prior to automatic stop and permanent delete operations occurring. If you subscribe to Oracle Cloud Infrastructure Alerts and Notifications, you also will receive email notifications.

Note:

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

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. Promotion of Always Free to a paid Autonomous Database is supported only if the Database version for the Always Free Autonomous Database is Oracle Database 19c.
Note:
For details on upgrading your Always Free APEX Service to an Oracle APEX Application Development paid instance, see Upgrading Always Free APEX Service to a Paid Version.

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

1. Open the Oracle Cloud Infrastructure Console by clicking the ➔ next to Oracle Cloud.
2. From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
3. On the Autonomous Databases page select an Always Free Autonomous Database from the links under the Display Name column.
4. On the Autonomous Database Details page, from the More Actions drop-down list select Upgrade Instance to Paid.
5. Click Upgrade Instance to Paid.

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

Security and Authentication in Oracle Autonomous Database

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

Note:
Oracle Autonomous Database supports the standard security features of the Oracle Database including privilege analysis, network encryption, centrally managed users, secure application roles, transparent sensitive data protection, and others. Additionally, Oracle Autonomous Database adds Label Security, Database Vault, Data Safe, and other advanced security features at no additional cost.

Topics
• Configuration Management
• Data Encryption
• Data Access Control
• Auditing
• Assessing the Security of Your Database and its Data
Configuration Management

The Oracle Autonomous Database provides standard, hardened security configurations that reduce the time and money managing configurations across your databases. Security patches and updates are done automatically, so you don't spend time, money, or attention to keeping security up to date. These capabilities protect your databases and data from costly and potentially disastrous security vulnerabilities and breaches.

Data Encryption

Oracle Autonomous Database uses always-on encryption that protects data at rest and in transit. Data at rest and in motion is encrypted by default. Encryption cannot be turned off.

Encryption of Data at Rest

Data at rest is encrypted using TDE (Transparent Data Encryption), a cryptographic solution that protects the processing, transmission, and storage of data. Using AES256 tablespace encryption, each database has its own encryption key, and any backups have their own different encryption keys.

By default, Oracle Autonomous Database creates and manages all the master encryption keys used to protect your data, storing them in a secure PKCS 12 keystore on the same systems where the database resides. If your company security policies require, Oracle Autonomous Database can instead use keys you create and manage in the Oracle Cloud Infrastructure Vault service. For more information, see About Master Encryption Key Management on Autonomous Database.

Additionally, customer-managed keys can be rotated when needed in order to meet your organization's security policies.

Note: When you clone a database, the new database gets its own new set of encryption keys.

Encryption of Data in Transit

Clients (applications and tools) connect to Oracle Autonomous Database using supported protocols including SQL*Net, JDBC, and ODBC.

TCPS (Secure TCP) database connection services use the industry-standard TLS 1.2 (Transport Layer Security) protocol for connections and symmetric-key data encryption.

- With mTLS connections, Oracle Autonomous Database users download a connection wallet that contains all necessary files for a client to connect using TCPS. Distribute this wallet only to those users who need and are permitted to have database access. The client-side configuration uses information in the wallet to perform symmetric-key data encryption.
- Autonomous Database by default supports Mutual TLS (mTLS) connections. You have the option to configure an Autonomous Database instance to allow both mTLS and TLS connections. Using TLS connections, some clients, such as JDBC
Thin Driver clients, do not need to download a wallet if you use a TLS connection string and TLS is enabled for the Autonomous Database instance.

See Secure Connections to Autonomous Database for more information.

### Data Access Control

Securing access to your Oracle Autonomous Database and your data uses several different kinds of access control:

- **Client Access Control**
- **Database User Access Control**
- **Oracle Cloud User Access Control**

#### Client Access Control

**Network Access Control**

You define network access control when you set up and configure your Oracle Autonomous Database. There are two options to consider.

- **Private Endpoints and Security Lists**: This is the recommended option. Create your Oracle Autonomous Database in your virtual cloud network (VCN) using private endpoints. You control access to your database using security lists and network security groups allowing you to specify who can create connections to your database.

  For detailed information about creating these resources, see Configuring Network Access with Private Endpoints.

- **Public Endpoints and Access Control Lists**: Create your Oracle Autonomous Database using public endpoints allowing access from any client with client credentials. You control access to your database using network access control lists (ACLs) allowing you to specify IP addresses, CIDR blocks, or VCNs that can connect to your database. Public IPs are easier to discover and attack, and Private Endpoints are recommended where possible.

  For detailed information about setting up an ACL, see Configure Access Control Lists for an Existing Autonomous Database Instance.

#### Database User Access Control

The Oracle Autonomous Database is configured with an administrative account, ADMIN, that is used to create and manage other database accounts. Oracle Autonomous Database provides a robust set of features and controls including system and object privileges and roles. User profiles allows you to customize password policies to define and implement a secure database user access strategy.
For basic information about standard user management, see User Accounts in *Oracle Database Concepts*. For detailed information and guidance, see Managing Security for Oracle Database Users in *Oracle Database Security Guide*.

If your database user access strategy demands more controls than are provided by standard user management, you can configure your Oracle Autonomous Database to use Database Vault to meet more rigorous requirements.

**Using Microsoft Active Directory to Manage Database Users**

If you use Microsoft Active Directory as a user repository, you can configure your database to authenticate and authorize Microsoft Active Directory users. This integration enables you to consolidate your user repository while still implementing a rigorous database user access strategy, regardless of whether you use standard user management or Database Vault.

For more information about integrating Microsoft Active Directory with your databases, see *Use Microsoft Active Directory with Autonomous Database*.

**Database Vault**

Oracle Database Vault comes preconfigured and ready to use. You can use its powerful security controls to restrict access to application data by privileged database users, reducing the risk threats, and addressing common compliance requirements.

You configure controls to block privileged account access to application data and control sensitive operations inside the database. You configure trusted paths to add additional security controls to authorized data access, database objects, and database commands. Database Vault secures existing database environments transparently, eliminating costly and time consuming application changes.

Before using Database Vault, be sure to review *Use Oracle Database Vault with Autonomous Database* to gain an understanding of the impact of configuring and enabling Database Vault.

For detailed information on implementing Database Vault features, refer to *Oracle Database Vault Administrator’s Guide*.

**Oracle Cloud User Access Control**

You use Identity and Access Management (IAM) services to control the privileges of your Oracle Cloud users by specifying the actions those users can perform on your Oracle Autonomous Database.

The IAM service provides several kinds of components to help you define and implement a secure cloud user access strategy:

- **Compartment**: A collection of related resources. Compartments are a fundamental component of Oracle Cloud Infrastructure for organizing and isolating your cloud resources.
- **Group**: A collection of users who all need the same type of access to a particular set of resources or compartment.
- **Dynamic Group**: A special type of group that contains resources that match rules that you define. Thus, the membership can change dynamically as matching resources are created or deleted.
• **Policy**: A group of statements that specify who can access which resources, and how. Access is granted at the group and compartment level, which means you write a policy statement that gives a specific group a specific type of access to a specific type of resource within a specific compartment.

Of these, the policy is the primary tool you use to control access because it provides the "Who", "How", "What" and "Where" of a single access constraint. A policy statement has this format:

The format of a policy statement is:

```
Allow
group <group-name>
to <control-verb>
<resource-type>
in compartment <compartment-name>
```

• **group <group-name>** specifies the "Who" by providing the name of an existing IAM group.

• **to <control-verb>** specifies the "How" using one of these predefined control verbs:
  
  – **inspect**: the ability to list resources of the given type, without access to any confidential information or user-specified metadata that may be part of that resource.
  
  – **read**: inspect plus the ability to get user-specified metadata and the actual resource itself.
  
  – **use**: read plus the ability to work with existing resources, but not to create or delete them. Additionally, "work with" means different operations for different resource types.
  
  – **manage**: all permissions for the resource type, including creation and deletion.

• **<resource-type>** specifies the "What" using a predefined resource-type. The resource-type values for infrastructure resources are:
  
  – autonomous-databases
  
  – autonomous-backups

You may create policy statements that refer to the **tag-namespaces** resource-type value if tagging is used in your tenancy.

• **in compartment <compartment-name>** specifies the "Where" by providing the name of an existing IAM compartment.

For information about how the IAM service and its components work and how to use them, see **Overview of Oracle Cloud Infrastructure Identity and Access Management**. For quick answers to common questions about IAM, see the **Identity and Access Management FAQ**.

**Auditing**

Oracle Autonomous Database provides robust auditing capabilities that enable you to track who did what on the service and on specific databases. Comprehensive log data allows you to audit and monitor actions on your resources, which helps you to meet your audit requirements while reducing security and operational risk.

• **Auditing Service Level Activities**

• **Auditing Database Activities**
Auditing Service Level Activities

All actions Oracle Cloud users perform on the resources that make up your deployment of Oracle Autonomous Database are logged by the Audit service, regardless of the interface used: the Oracle Cloud Infrastructure Console, REST API, Command Line Interface (CLI), Software Development Kits (SDK) and so on.

You can use the Audit service to perform diagnostics, track resource usage, monitor compliance, and collect security-related events. For more information about the Audit service, see Overview of Audit in Oracle Cloud Infrastructure Documentation.

Additionally, when users perform operations on your Oracle Autonomous Database, the database publishes events to the Oracle Cloud Events service. The Oracle Cloud Events service allows you to create rules to capture these events and perform actions.

For more information about how the Events service works and how to set up the rules and actions it uses, see Overview of Events. For listings of the Oracle Autonomous Database operations that generate events, see Autonomous Database Event Types.

Auditing Database Activities

Oracle Autonomous Database configures the autonomous databases you create to use the unified auditing feature of Oracle Database.

This feature captures audit records from the following sources and gathers them in a single audit trail in a uniform format:

- Audit records (including SYS audit records) from unified audit policies and AUDIT settings
- Fine-grained audit records from the DBMS_FGA PL/SQL package
- Oracle Database Real Application Security audit records
- Oracle Recovery Manager audit records
- Oracle Database Vault audit records
- Oracle Label Security audit records
- Oracle Data Mining records
- Oracle Data Pump
- Oracle SQL*Loader Direct Load

Audit information is retained for up to 14 days, after which it is automatically purged. To retain audit information for longer, and to easily analyze and report on database activity, use Oracle Data Safe (included with your Oracle Autonomous Database subscription).

See About Auditing Autonomous Database for more information.

Assessing the Security of Your Database and its Data

Oracle Autonomous Database integrates with Oracle Data Safe to help you assess and secure your databases.

Oracle Data Safe helps you understand the sensitivity of your data, evaluate risks to data, mask sensitive data, implement and monitor security controls, assess user
security, monitor user activity, and address data security compliance requirements in your databases.

You use Oracle Data Safe to identify and protect sensitive and regulated data your Oracle Autonomous Database by registering your database with Data Safe. Then, you use the Data Safe console directly from the Details page of your database.

For more information about using Data Safe, see Safeguard Your Data with Data Safe on Autonomous Database.

Regulatory Compliance Certification

Oracle Autonomous Database meets a broad set of international and industry-specific compliance standards, including:

- HIPAA - Health Insurance Portability and Accountability Act
- ISO/IEC 27001:2013 - International Organization for Standardization 27001
- SOC 1 - System and Organization Controls 1
- SOC 2 - System and Organization Controls 2
- ISO 15408 - Common Criteria at EAL2

For more information and a complete list of certifications, see Oracle Cloud Compliance.

Provision an Autonomous Database

Provisioning a new Autonomous Database is easy. From the Oracle Cloud Infrastructure Console, on the Autonomous Databases page choose your region, select a compartment, and click Create Autonomous Database. This shows the Create Autonomous Database dialog.

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

After you create an Autonomous Database:

- (Optional) Reset the administrator password:
  When you create an Autonomous Database you are required to set the administrator password. If you want to change the administrator password or if you need to unlock the administrator account, see Manage the Administrator Account on Autonomous Database.

- (Optional) Add Oracle Machine Learning component users. See Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database for more information.

- (Optional) Download Oracle Analytics Desktop and add a connection to Autonomous Database. See Create Dashboards and Reports to Analyze and Visualize Your Data.
Use Sample Data Sets in Autonomous Database

For users who want to start using the service without creating their own tables, Autonomous Database provides the read-only Sales History and Star Schema Benchmark data sets.

These data sets are provided as Oracle Database schemas SH and SSB respectively. Any user can query these data sets without any manual configuration.

**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 Database 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 Database 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 Database.

Build Reports and Dashboards with Analytics in Autonomous Database

Working with Oracle Analytics Cloud

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

For more information, see Use Oracle Analytics Cloud with Autonomous Database.

Working with Oracle Analytics Desktop

You can use Oracle Analytics Desktop with Autonomous Database. Just connect to Autonomous Database, select the elements that you’re interested in, and let Oracle Analytics Desktop find the best way to visualize it. Choose from a variety of visualizations to look at data in a specific way.

For more information see Use Oracle Analytics Desktop with Autonomous Database.

Use Oracle Machine Learning Notebooks with Autonomous Database

Oracle Machine Learning Notebooks 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 Notebooks:

• Allows collaboration among data scientists, developers, business users
• Leverages the scalability and performance of Oracle Platform and its Cloud Services

To use Oracle Machine Learning Notebooks with Autonomous Database you need to add or create Oracle Machine Learning Notebook users and then access Oracle Machine Learning notebooks:

• **OML User Management** lets the Admin (user with administrative privileges) create and modify Oracle Machine Learning user accounts. See Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database for details on accessing OML User Management.

• **OML Application** – application users access Oracle Machine Learning Notebooks to create, view, and share notebooks for data analytics, data visualization, and other Oracle Machine Learning Notebooks tasks. Users access Oracle Machine Learning Notebooks by clicking the Oracle Machine Learning Home icon on the User Administration page or from the account details mailed to a new user. See Work with Oracle Machine Learning Notebooks for Data Access, Analysis, and Discovery for details on using the OML application.
Autonomous Database Tutorials

Provides links to Autonomous Database tutorials.

Autonomous Database 15 Minute Quick Start

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

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

Load and Analyze Your Data with Autonomous Database

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

- Provision a new Autonomous Database instance on Shared Infrastructure
- Run Queries on the sample data sets
- Upload files to the Oracle Cloud Infrastructure Object Storage, create sample tables, load data into them from files on the OCI Object Storage
- Query files on the Oracle Cloud Infrastructure Object Storage directly without loading them to your database
- Visualize your data using Oracle Analytics Desktop

Important Tools for Everyone Using Oracle Autonomous Database

Introduces the suite of data tools built into Oracle Autonomous Database.

- Provision an Oracle Autonomous Database instance
- Create the database user that you will use in the next lab
- Update the user's profile to grant access to load and store data
- Log in as the user
- Familiarize yourself with the suite of built-in database tools in Oracle Autonomous Database
- Load data
• Learn how to use the Data Transforms tool to correct data errors
• Create a business model
• Generate data insights
• Use the Catalog tool

Manage and Monitor Autonomous Database

Connect using secure wallets and monitor your Autonomous Database instances.

Manage and Monitor Autonomous Database Workshop
• Get comfortable with Oracle's public cloud services
• Provision a new Autonomous Database instance on Shared Exadata Infrastructure
• Run sample queries against the sample data sets
• Load data from the object store
• Query external data from the object store
• Scale an Autonomous Database instance

Produce Your Company's Best Picture with Converged Database Analytics on Autonomous Database

Learn about Autonomous Database and to deliver high value solutions using Oracle Cloud data platform services in the context of a company that provides a movie streaming service.

Produce Your Company's Best Picture with Converged Database Analytics - Deep Dive Workshop

Get Help, Search Forums, and Contact Support

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

Post Questions on Forums

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

Cloud Customer Connect Forums

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

• Autonomous Data Warehouse
• Autonomous Transaction Processing
• Oracle Cloud Infrastructure and Platform
Stack Overflow Knowledge Forum

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

Questions tagged [oracle-autonomous-db]

Open a Support Ticket

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

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

Use the Oracle Cloud Infrastructure Console to Create a Support Ticket

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

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

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

Availability Service Level Objectives (SLOs)

This topic describes the Service Level Objectives (SLOs) for Oracle Autonomous Database on Shared Exadata Infrastructure.

Recovery Time Objective Service Level Objectives

Oracle Autonomous Database on Shared Exadata Infrastructure is engineered to return an application online following an unplanned outage or a planned maintenance activity within single-digit seconds. The following table outlines the target Recovery Time Objective (RTO) SLOs for different failure events.
### Event

<table>
<thead>
<tr>
<th>Event</th>
<th>Recovery Time Objective (RTO) Service Level Objective</th>
<th>Recovery Point Objective (RPO) Service Level Objective, Maximum Possible Data Loss</th>
</tr>
</thead>
</table>
| Events requiring failover to a standby database when Autonomous Data Guard is enabled, such as:  
  - Data corruptions  
  - Full database failures  
  - Complete storage failures  
  - Availability Domain or Region failures | Local Standby Database two (2) minutes  
  Cross-Region Standby Database fifteen (15) minutes | The maximum possible data loss is 1 minute.  
  See Autonomous Data Guard Recovery Time Objective (RTO) and Recovery Point Objective (RPO) for more information. |
| In the case of an Autonomous Database without a standby database, with events that require restoring from backup, such as:  
  - Data corruptions  
  - Full database failures  
  - Complete storage failures | Based on the size of the database:  
  1 hour + (1 hour per 5 TB)  
  For example, a 5 TB database has a Recovery Time Objective (RTO) of up to 2 hours. | The maximum possible data loss is 1 minute. |
| In the case of an Autonomous Database without a standby database, with events that require restoring from backup, such as:  
  - Availability Domain failures, in regions with more than one Availability Domain  
  See Regions and Availability Domains for information on Availability Domains. | 48 hours | The maximum possible data loss is 1 minute. |
Connecting to Autonomous Database

Describes methods to securely connect to Autonomous Database.

**Topics**

- About Connecting to an Autonomous Database Instance
- Connect to Autonomous Database Using a Client Application
- Download Database Connection Information
- Connect to Autonomous Database Using Oracle Database Tools
- Connect with Built-in Oracle Database Actions
- Connect with JDBC Thin Driver
- Preparing for Oracle Call Interface Connections
- Predefined Database Service Names for Autonomous Database
- Connect with Oracle Analytics Cloud
- Connection and Networking Options and Features
- Use Database Links with Autonomous Database

**About Connecting to an Autonomous Database Instance**

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

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

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

- **JDBC Thin Driver**: for Java applications, the JDBC Thin Driver is a pure Java driver. Many applications, including Oracle SQL Developer support JDBC Thin Driver connections.

- **JDBC OCI**: which is used by Java language applications. JDBC OCI adds a layer over Oracle Call Interface for Java applications. The Oracle SQLcl command-line interface uses JDBC OCI.

- **Oracle Call Interface (OCI)**: used by many applications written in C language. Examples that use Oracle Call Interface include Oracle utilities such as Oracle SQL*Plus, SQL*Loader, and Oracle Data Pump.

- **ODBC Drivers**: used by applications running on Microsoft Windows, that are layered over Oracle Call Interface (OCI).
Third-party products and custom applications may use any of these connection types.

Secure Connections to Autonomous Database

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

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

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

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

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

About Mutual TLS (mTLS) Authentication

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

Mutual TLS authentication requires that the client downloads or obtains a trusted client CA certificate for connecting to an Autonomous Database instance. Autonomous Database then uses the certificate to authenticate the client. This provides increased security and specifies the clients that can communicate with an Autonomous Database instance.
Certification authentication with Mutual TLS uses an encrypted key stored in a wallet on both the client (where the application is running) and the server (where your database service on the Autonomous Database is running). The key on the client must match the key on the server to make a connection. A wallet contains a collection of files, including the key and other information needed to connect to your Autonomous Database instance. All communications between the client and the server are encrypted.

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

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

**About TLS Authentication**

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

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

TLS connections are only supported in commercial regions. Government regions only allow mTLS connections to an Autonomous Database instance.

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

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

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

Connecting to Autonomous Database Through a Firewall

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

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

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

```sql
db2022adb_high = (description = (address=(protocol=tcps)
   (port=1522)
   (host=adb.example.oraclecloud.com))
   (connect_data=(service_name=example_high.adb.oraclecloud.com))
   (security=(ssl_server_cert_dn="CN=adb.example.oraclecloud.com,
     OU=Oracle BMCS US,O=Oracle Corporation,L=Redwood City,ST=California,C=US")))
```
Your firewall must allow access to servers within the .oraclecloud.com domain using port 1522. To connect to Autonomous Database, depending upon your organization's network configuration, you may need to use a proxy server to access this port or you may need to request that your network administrator open this port.

Using Application Continuity

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

Note:

By default Application Continuity is disabled.

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

Connect to Autonomous Database Using a Client Application

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

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

Topics

- About Connecting to Autonomous Database Using a Client Application
- Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections with Wallets (mTLS)
- Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections Using TLS Authentication
- Prepare for JDBC Thin Connections
- Connect Microsoft .NET, Visual Studio Code, and Visual Studio with a Wallet (mTLS)
- Connect Microsoft .NET, Visual Studio Code, and Visual Studio without a Wallet
- Connect Python, Node.js, and other Scripting Languages (mTLS)
- Connect Python, Node.js, and other Scripting Languages without a Wallet

About Connecting to Autonomous Database Using a Client Application

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

Consult your application documentation for details about how your application connects to Oracle.
The following steps describe the process of connecting to Autonomous Database using a client application:

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

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

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

See the details for each driver type for information on the steps required to prepare your application to connect to Autonomous Database.

3. Within your application, set up the connection.

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

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

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

New Oracle Client Installation

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

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

1. Install Oracle Client software on your computer. 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
   ```

   The changed value on UNIX/Linux is:

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

   The changed value for Windows is:

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

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

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

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

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

   For example on Windows:

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

### Connections with an HTTP Proxy

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

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

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

   SQLNET.USE_HTTPS_PROXY=on

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

   ADB1_high =
   (description=
     (address=
       (https_proxy=proxyhostname) (https_proxy_port=80)
       (protocol=tcps) (port=1522) (host=adb.example.oraclecloud.com)
     )
     (connect_data=(service_name=adb1_high.adb.oraclecloud.com)
     )
     (security=(ssl_server_cert_dn="adb.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 Protocol Address Section.
### Updating an Existing Oracle Client Installation

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

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

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

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

### Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections

#### Using TLS Authentication

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

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

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

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

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

   ```
   cd /home/myuser/instantclient_19_14
   ln -s libclntsh.so.19.1 libclntsh.so
   ```
If there is no other Oracle software on your system that will be impacted, add Instant Client to the runtime link path. For example:

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

```
sudo ldconfig
```

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

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

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

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

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

### 3. Run Your Application

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

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

#### c. Run your application.

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

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

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

## Prepare for JDBC Thin Connections

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

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

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

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

Set JVM Networking Properties

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

For the Java Virtual Machine (JVM) address cache, any address resolution attempt caches the result whether it was successful or not, so that subsequent identical requests do not have to access the naming service. The address cache properties allow you to tune how the cache operates. In particular, the networkaddress.cache.ttl value specifies the number of seconds a successful name lookup is kept in the cache. A value of -1, the default value, indicates a “cache forever” policy, while a value of 0 (zero) means no caching.

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

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

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

- Set the following property in your application code:

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

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


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

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


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

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

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

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

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

ODP.NET connectivity to Oracle Autonomous Database supports mutual TLS (mTLS) with wallets as well as TLS. TLS connections without wallets are supported with managed ODP.NET and ODP.NET Core starting with version 19.13 and 21.4. See Connect Microsoft .NET, Visual Studio Code, and Visual Studio without a Wallet for information on using TLS connections.

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


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

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

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


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

- Oracle Data Access Components - .NET Downloads
• NuGet Gallery

• Visual Studio Code Marketplace

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

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

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

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

2. Enable TLS connections on your Autonomous Database instance. See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for details.

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

Notes for the TLS connection string:

• The TLS connection string uses quotation marks around the distinguished name. If you store the TLS connection string in a .NET string, add a backslash escape sequence before each quotation mark (for example, \\") This allows .NET to recognize the quotation mark as part of the TLS connection string.

• Verify that the connection string includes (SECURITY=(SSL_SERVER_DN_MATCH=TRUE)) to ensure that the client matches the server DN. If not specified, add this to the connect string. For example:

   (description=
   (retry_count=20)(retry_delay=3)(address=(protocol=tcps)(port=1521)
   (host=HOSTNAME))(connect_data=(service_name=SERVICE_NAME))
   (security=(SSL_SERVER_DN_MATCH=TRUE)
   (ssl_server_cert_dn=DISTINGUISHED_NAME)))

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

Connect Python, Node.js, and other Scripting Languages (mTLS)

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

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

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

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

For details on connecting with TLS without a wallet, see Connect Python, Node.js, and other Scripting Languages without a Wallet.

For additional details, see the videos:

- Connect Node.js Apps to Autonomous Transaction Processing
- Connect Python Apps to Autonomous Transaction Processing

Install the Language Driver and Client Libraries

To connect to Oracle Autonomous Database 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 librclntsh.so.18.1 librclntsh.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:

```bash
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:

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

**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.
   - **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 Autonomous Database. You obtain a zip file containing client security credentials and network configuration settings required to access your database. You must protect this file and its contents to prevent unauthorized database access. Obtain the client security credentials file as follows:
   - **ADMIN user**: Click **DB Connection**. See Download Client Credentials (Wallets).
   - **Other user (non-administrator)**: Obtain the Oracle Wallet from the administrator for your Autonomous Database.

2. Extract the client credentials (wallet) files:
a. Unzip the client credentials zip file.

b. If you are using Instant Client, make a `network/admin` subdirectory hierarchy under the Instant Client directory if necessary. Then move the files to this subdirectory. For example depending on the architecture or your client system and where you installed Instant Client, the files should be in the directory:

   C:\instantclient_12_2\network\admin

   or

   /home/myuser/instantclient_18_5/network/admin

   or

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

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

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

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

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

Run Your Application

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

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

3. Run your application.
Connect Python, Node.js, and other Scripting Languages without a Wallet

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

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

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

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

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

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

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

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

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

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

Chapter 2
Connect to Autonomous Database Using a Client Application
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](#).
   • **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](#).

3. If you have not already done so, enable TLS connections on your Autonomous Database instance.
   See [Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication](#) for details.

4. Run Your Application
   a. Update your application to connect using your database username, your password, and the Oracle Net connect name given in the unzipped `tnsnames.ora` file. For example, `user`, `adb_user`, `password`, and `db2022adb_low` as the connect string.
   b. Alternatively, change the connect string in `tnsnames.ora` to match the string used by your application.
   c. Run your application.

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

Oracle client credentials (wallet files) are downloaded from an Autonomous Database instance by a service administrator. If you are not an Autonomous Database administrator and your application requires a wallet to connect, then your administrator should provide you with the client credentials. You can also view TNS names and connection strings for your database.

Topics

• Download Client Credentials (Wallets)
• Wallet README File
• View TNS Names and Connection Strings for an Autonomous Database Instance

Download Client Credentials (Wallets)

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

Note:

The password you provide when you download the wallet protects the downloaded Client Credentials wallet.

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

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

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

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

To download client credentials from the Oracle Cloud Infrastructure Console:

1. Navigate to the Autonomous Database details page.
2. Click 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).

> **Note:**

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

4. Click **Download Wallet**.

5. In the **Download Wallet** dialog, enter a wallet password in the **Password** field and confirm the password in the **Confirm Password** field.

6. Click **Download** to save the client security credentials zip file.

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

   You must protect this file to prevent unauthorized database access.

**To download client credentials from Database Actions:**

First, access Database Actions as the ADMIN user. See [Access Database Actions as ADMIN](#) for more information.

1. Access Database Actions as the ADMIN user. See [Access Database Actions as ADMIN](#) for more information.

2. On the Database Actions Launchpad, under **Administration**, select **Download Client Credentials (Wallet)**.

3. On the **Download Client Credentials (Wallet)** page, enter a wallet password in the **Password** field and confirm the password in the **Confirm Password** field.

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

> **Note:**

When you use Database Actions to download a wallet there is no **Wallet Type** option on the **Download Client Credentials (Wallet)** page and you always download an instance wallet. If you need to download the regional wallet use **DB Connection** on the Oracle Cloud Infrastructure Console.

The zip file includes the following:

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

Notes for wallet files and the wallet password:

- To invalidate database client certification keys associated with a wallet, see [Rotate Wallets with Immediate Rotation](#).
- Wallet files, along with the Database user ID and password provide access to data in your database. Store wallet files in a secure location. Share wallet files only with authorized users. If wallet files are transmitted in a way that might be accessed by unauthorized users (for example, over public email), transmit the wallet password separately and securely.
- For better security, Oracle recommends using restricted permissions on wallet files. This means setting the file permissions to 600 on Linux/Unix. Similar restrictions can be achieved on Windows by letting the file owner have Read and Write permissions while all other users have no permissions.
- Autonomous Database uses strong password complexity rules for all users based on Oracle Cloud security standards. For more information on the password complexity rules see [Create Users on Autonomous Database - Connecting with a Client Tool](#).
- The README file that contains wallet expiration information is not available in wallet zip files that were downloaded before April 2020.
- Starting six weeks before the wallet expiration date Autonomous Database sends notification emails each week, indicating the wallet expiration date. These emails provide notice before your wallet expires that you need to download a new wallet. You will receive these notification emails only if there is a connection that uses a wallet that is about to expire.

You can also use the WalletExpirationWarning event to be notified when a wallet is due to expire. You will receive these notification events only if you are subscribed to Critical events and there is a connection that uses a wallet that is about to expire. See [About Chapter 2](#) Download Database Connection Information.
Events Based Notification and Automation on Autonomous Database for more information.

Wallet README File

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

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

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

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

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

<table>
<thead>
<tr>
<th>Tool or Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Machine Learning User Notebooks</td>
<td>Oracle Machine Learning notebooks provide easy access to Oracle’s parallelized, scalable in-database implementations of a library of Oracle Advanced Analytics’ machine learning algorithms (classification, regression, anomaly detection, clustering, associations, attribute importance, feature extraction, times series, etc.), SQL, PL/SQL and Oracle’s statistical and analytical SQL functions. Access Link: provides the link to use Oracle Machine Learning User Notebooks. See Work with Oracle Machine Learning Notebooks for Data Access, Analysis, and Discovery for more information.</td>
</tr>
<tr>
<td>SODA Drivers</td>
<td>Simple Oracle Document Access (SODA) is a set of APIs that let you work with JSON documents managed by the Oracle Database without needing to use SQL. SODA drivers are available for REST, Java, Node.js, Python, PL/SQL, and C. Access Link: provides the link to download the SODA drivers. See Work with Simple Oracle Document Access (SODA) in Autonomous Database for more information.</td>
</tr>
</tbody>
</table>

**Notes for wallet README file:**

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

### View TNS Names and Connection Strings for an Autonomous Database Instance

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

**Note:**

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

Perform the following steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the Display Name column.
To view the TNS names and connection strings, do the following:

1. On the Autonomous Database details page, click DB Connection.
   
   By default this shows the Mutual TLS connection information in a table with the TNS names and connection strings for the Autonomous Database instance.

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

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

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

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

You will need the client credentials and connection information to connect to your database. The client credentials include the wallet.

Download Client Credentials (Wallet)

To download your client credentials, select the type of wallet, then click Download Wallet. You will be asked to create a password for the wallet.

Wallet Type

Instance Wallet

Download Wallet Rotate Wallet

Wallet last rotated:

Connection Strings

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

TLS Authentication

TLS

TNS Name  Connection String

databasename _high  (description= (retry_count=20)(retry_delay=3)(address=(protocol=tcps)(port=1521)(host=adb-preprod.us-phoenix-1.oraclecloud.com))(connect_data=(service_name=usac8x4d_databasename_high.adb.oraclecloud.com)(security=(ssl_server_cert_dn="CN=adwc-preprod.uscom-east-1.oraclecloud.com, OU=Oracle BMCS US, O=Oracle Corporation, L=Redwood City, ST=California, C=US"))))

databasename _low  ...dwood City, ST=California, C=US")))

databasename _medium  ...dwood City, ST=California, C=US")))

Showing 3 Items

Close
Connect to Autonomous Database Using Oracle Database Tools

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

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

Topics

- Connect Oracle SQL Developer with a Wallet (mTLS)
- Connect Oracle SQL Developer without a Wallet
- Connect Oracle SQL Developer (earlier than Version 18.2) with a Wallet (mTLS)
- Connect SQL*Plus with a Wallet (mTLS)
- Connect SQL*Plus without a Wallet
- Connect Oracle SQLcl Cloud with a Wallet (mTLS)
- Connect Oracle SQLcl Cloud without a Wallet

Connect Oracle SQL Developer with a Wallet (mTLS)

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

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

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

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

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

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

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

---

**Note:**

Versions of SQL Developer before 18.2 require that you enter a **Keystore Password**. For more information, see Connect Oracle SQL Developer (earlier than Version 18.2) with a Wallet (mTLS).
4. Click **Connect** to connect to the database.

**Note:**

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

---

**Connect Oracle SQL Developer without a Wallet**

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

**Note:**

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

---

To create a new TLS connection to Autonomous Database:

1. Copy a connection string for the Autonomous Database.

   To connect with TLS authentication copy a TLS connection string. On the Database Connection page, under **TLS Authentication**, select **TLS** to view the connection strings for connecting with TLS authentication.
2. Start Oracle SQL Developer and in the connections panel, right-click **Connections** and select **New Database Connection**.

3. Choose the Connection Type **Custom JDBC**.

4. Enter the following information:
   - **Name**: Enter the name for this connection.
   - **Username**: Enter the database username. You can either use the default administrator database account **ADMIN** provided as part of the service or create a new schema, and use it.
   - **Password**: Enter the password for the database user.
   - **Connection Type**: Select **Custom JDBC**.
   - **Custom JDBC URL**: Enter the following:
     
     jdbc:oracle:thin:@ followed by the connection string you copied in step one.

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

```java
jdbc:oracle:thin:@(description= (retry_count=20)(retry_delay=3)
(address=(protocol=tcps)
(port=1521)(host=adb-preprod.us-phoenix-1.oraclecloud.com))
(connect_data=(service_name=u9adutfb2ba8x4d_databasename_medium.adb.oraclecloud.com))
(security=(ssl_server_cert_dn="CN=adwc-preprod.region.oraclecloud.com,
OU=Oracle BMCS US, O=Oracle Corporation, L=Redwood City, ST=California, C=US"))
```

When you copy the connection string, the values for **region** and **databasename** are for your Autonomous Database instance.
5. Click **Connect** to connect to the database.

**Note:**

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

---

**Connect Oracle SQL Developer (earlier than Version 18.2) with a Wallet (mTLS)**

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 Database using an Oracle Wallet and this version contains enhancements for key Autonomous Database 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 Database.

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

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

To create a new connection to the Autonomous Database, 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:

   ![New/Select Database Connection](image)

   **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 Oracle SQL Developer with a Wallet (mTLS)*.

   - **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 Database. 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 SQL*Plus with a Wallet (mTLS)

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

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

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

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

For example:

```sql
sqlplus adb_user@db2022adb_medium
```

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

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

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

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

SQL>
```

Notes:

- The Oracle Wallet is transparent to SQL*Plus because the wallet location is specified in the sqlnet.ora file. This is true for any Oracle Call Interface (OCI), ODBC, or JDBC OCI connection.

- If you are connecting to a database using Microsoft Active Directory credentials, then connect using an Active Directory user name in the form of "AD_domain\AD_username" (double quotes must be included), and Active Directory user password. See Use Microsoft Active Directory with Autonomous Database for more information.
Connect SQL*Plus without a Wallet

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

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

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

1. Prepare for Oracle Call Interface (OCI) connections.
   You can use TLS authentication without a wallet in SQL*Plus if you are using the following client versions:
   - Oracle Instant Client/Oracle Database Client 19.13 - only on Linux x64
   - Oracle Instant Client/Oracle Database Client 19.14 (or later) and 21.5 (or later) - all platforms
   See Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections Using TLS Authentication for more information.

2. Copy a connection string for the Autonomous Database.
   To connect with TLS authentication copy a TLS connection string. On the Database Connection page, under TLS Authentication, select TLS to view the connection strings for connecting with TLS authentication.
   See View TNS Names and Connection Strings for an Autonomous Database Instance for information on viewing and copying connection stings.
   See Predefined Database Service Names for Autonomous Database for information on the different databases services for each connection string.

3. Connect using a database user and password, and provide the connection string you copied in Step 2.
   On UNIX/Linux start sqlplus with the connection string, enclosed in quotes on the command line, as follows:
   sqlplus username/password@'my_connect_string'

   For example (for clarity line breaks added):
   sqlplus adb_user/password@'(description= (retry_count=20)(retry_delay=3)
   (address=(protocol=tcps)
   (port=1521)(host=adb.region.oraclecloud.com))
   (connect_data=(service_name=u9adutfb2ba8x4d_database_medium.adb.oraclecloud.com))
   (security=(ssl_server_cert_dn="CN=adwc-preprod.uscom-east-1.oraclecloud.com, OU=Oracle BMCS US, O=Oracle Corporation, L=Redwood City, ST=California, C=US")))'
On Windows, start `sqlplus` with the database user and password with the copied connection string, as follows (as compared to UNIX/Linux, on Windows do not surround the connection string with quotes):

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

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

```
sqlplus adb_user/password@(description= (retry_count=20) (retry_delay=3) (address=(protocol=tcps) (port=1521) (host=adb.region.oraclecloud.com)) (connect_data=(service_name=u9adutfb2ba8x4d_database_medium.adb.oraclecloud.com)) (security=(ssl_server_cert_dn="CN=adwc-preprod.uscom-east-1.oraclecloud.com, O=Oracle BMCS US, C=US")))
```

**Note:**

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

## Connect Oracle SQLcl Cloud with a Wallet (mTLS)

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

You can use SQLcl version 4.2 or later with Autonomous Database. Download SQLcl from [oracle.com](http://oracle.com).

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

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

### SQLcl with Oracle Call Interface

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

```
sql -oci
```

SQLcl: Release 22.1 Production on Fri May 06 16:07:46 2022
When connecting using Oracle Call Interface, the Oracle Wallet is transparent to SQLcl.

**SQLcl with a JDBC Thin Connection**

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

1. Start SQLcl with the /nolog option.
   ```bash
   sql /nolog
   ```

2. Configure the SQLcl session to use your Oracle Wallet:
   ```sql
   SQL> set cloudconfig directory/client_credentials.zip
   ```

3. Connect to the database:
   ```sql
   SQL> connect username@servicename <password>
   ```

   For example:
   ```bash
   sql /nolog
   SQLcl: Release 22.1 Production on Fri May 06 14:48:26 2022
   ```

**SQLcl with a JDBC Thin Connection with an HTTP Proxy**

1. Start SQLcl with the /nolog option.
   ```bash
   sql /nolog
   ```

2. Configure the SQLcl session to use a proxy host and your Oracle Wallet:
   ```sql
   SQL> set cloudconfig -proxy=proxyhost:port directory/client_credentials.zip
   ```

3. Connect to the database:
   ```sql
   SQL> connect username@servicename <password>
   ```
For example:

```sql
sql /nolog
```

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

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

```sql
SQL> set cloudconfig -proxy=http://myproxyhost.com:80 /home/adb/Wallet_db2022.zip
```

```sql
SQL> connect adb_user@db2022adb_medium
```

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

**Note:**

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

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

For information on SQLcl, see Oracle SQLcl.

## Connect Oracle SQLcl Cloud without a Wallet

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

**Note:**

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

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

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

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

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

See [View TNS Names and Connection Strings for an Autonomous Database Instance](#) for information on viewing and copying connection strings.

See [Predefined Database Service Names for Autonomous Database](#) for information on the different databases services for each connection string.

2. Start SQLcl and connect to the database:

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

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

For example (for clarity line breaks added):

```
$ sql admin/password@'(description= (retry_count=20)(retry_delay=3)(address=(protocol=tcps)(port=1521)(host=adb.region.oraclecloud.com))(connect_data=(service_name=u9adutfb2ba8x4d_database_medium.adb.oraclecloud.com))(security=(ssl_server_cert_dn="CN=adwc-preprod.uscom-east-1.oraclecloud.com, OU=Oracle BMCS US, O=Oracle Corporation, L=Redwood City, ST=California, C=US"))'
```

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

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

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

SQL>

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

```
> sql /nolog
```

SQLcl: Release 21.2 Production on Fri Sep 17 10:15:01 2021
Copyright (c) 1982, 2021, Oracle. All rights reserved.

SQL> conn username/password@my_connect_string

For example (for clarity line breaks are added):

```
> sql admin/password@(description= (retry_count=20)(retry_delay=3)(address=(protocol=tcps)(port=1521)(host=adb.region.oraclecloud.com))(connect_data=(service_name=u9adutfb2ba8x4d_database_medium.adb.oracleclou d.com))(security=(ssl_server_cert_dn="CN=adwc-preprod.uscom-
```

---

Chapter 2

Connect to Autonomous Database Using Oracle Database Tools

2-37
Note:
If you are connecting to Autonomous Database using Microsoft Active Directory credentials, then connect using an Active Directory user name in the form of "AD_domain\AD_username" (double quotes must be included), and Active Directory user password. See Use Microsoft Active Directory with Autonomous Database for more information.

For information on SQLcl, see Oracle SQLcl.

Connect with Built-in Oracle Database Actions

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

Topics

• About Database Actions (SQL Developer Web)
• Access Database Actions as ADMIN
• Provide Database Actions Access to Database Users
• Access Database Actions, Oracle APEX, Oracle REST Data Services, and Developer Tools Using a Vanity URL

About Database Actions (SQL Developer Web)

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

These are the main features of Database Actions:

• Development features: SQL, Data Modeler, REST, JSON, Charts, Scheduling, Oracle Machine Learning, and Oracle APEX
• Data Tools features: Data Pump, Data Load, Catalog, Data Insights, Data Transforms, Data Analysis
• Administration features: Database Users, APEX Workspaces, Download Client Credentials, Set Resource Management Rules
• Monitoring features: Performance Hub and Database Monitor
• Downloads and Related Services: Download Oracle Instant Client, Download SODA Drivers, Access Restful Data Services (ORDS) and SODA, Access Oracle Machine Learning Restful Services

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

Access Database Actions as ADMIN

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

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

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

Note:

If your Autonomous Database is configured to use a Private Endpoint, then you can only access Database Actions from clients in the same Virtual Cloud Network (VCN).

To access Database Actions from the Oracle Cloud Infrastructure Console:

1. On the Autonomous Database Details page click Database Actions.
2. On the Database Actions Launchpad, select a card.
   For example, click SQL to use a SQL Worksheet. On the SQL Worksheet you can use the Consumer Group drop-down list to select the consumer group to run your SQL or PL/SQL code. See Executing SQL Statements in the Worksheet Editor for more information.

Provide Database Actions Access to Database Users

The ADMIN user can provide access to Database Actions to other database users.

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

• Use Database Actions to add a new user, if the user does not already exist and enable Web Access for the user. If the user already exists use similar steps but just enable Web Access for the schema.
• Provide the user with a URL to access Database Actions.
1. As the ADMIN user, access Database Actions and create a user with the required privileges.
   a. In Database Actions, click to show the available actions.
   b. In Database Actions, under Administration select Database Users.
   c. If the user you are adding does not already exist, click Create User.
   d. In the Create User area, on the User tab enter User Name and a Password and confirm the password.
   e. Select Web Access.
   f. In the Create User area, select Granted Roles tab and grant the appropriate roles to the user.
   g. Click Create User.

2. After adding a user and enabling Web Access, the ADMIN provides a user with the URL to access Database Actions, as follows:
   You have two options to supply the Database Actions URL to a user:
   • Supply the Wallet file to the user and inform the user that the Wallet README file includes the Database Actions URL. See Download Client Credentials (Wallets) for more information.
   • Copy the URL and provide it to the user with the following steps:
     a. Select the Autonomous Database.
     b. On the Autonomous Database Details page click Database Actions button.
     c. Copy the URL and replace everything after /ords/ with sql-developer.
        For example:
        https://dbname_id.adbregion.example.com/ords/sql-developer

3. Provide the user with this URL.
   To access Database Actions a user pastes the URL into their browser and then enters their Username and Password in the Sign-in dialog.
   See Create Users on Autonomous Database for information on adding database users.
   See Manage Users and User Roles on Autonomous Database - Connecting with Database Actions for more information.
   As an alternative, the ADMIN can provide Web Access for a user, REST enable a user, with SQL commands. As the ADMIN user run the following code:

   BEGIN
   ORD$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;
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 Database Actions.
- `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`.

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

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

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

To learn more, see the following:

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

Connect with JDBC Thin Driver

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

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

- **Mutual TLS (mTLS) Authentication**: requires either Oracle Wallet or Java KeyStore (JKS) where both the client and Autonomous Database authenticate each other.
  
The wallet and keystore files are included in the client credentials `.zip` file that is available by clicking DB Connection on the Oracle Cloud Infrastructure Console

- **TLS Authentication**: The client computer matches the server’s CA root certificate against the client’s list of trusted CAs. If the issuing CA is trusted, the client verifies that the certificate is authentic. This allows the client and Autonomous Database to establish the encrypted connection before exchanging any messages.

Topics

- JDBC Thin Connections with a Wallet (mTLS)
- JDBC Thin TLS Connections without a Wallet
JDBC Thin Connections with a Wallet (mTLS)

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

**Note:**

If you use TLS (instead of mTLS) for your connections using JDBC Thin Driver with JDK8u162 or higher, a wallet is not required.

TLS connections are enabled for the following network configurations:

- **Private endpoint access only**: network configuration with a private endpoint
- **Secure access from allowed IPs and VCNs only**: configuration with an Access Control List (ACL)

If your Autonomous Database is on a public endpoint without any ACL, you can add 0.0.0.0/0 as your CIDR ACL and enable TLS authentication. Adding 0.0.0.0/0 as your CIDR ACL is identical to having your Autonomous Database on public endpoint with no ACL.

See About TLS Authentication for more information.

Topics

- JDBC Thin Driver Connection Prerequisites Connections with Wallets (mTLS)
- Using a JDBC URL Connection String with JDBC Thin Driver and Wallets
- 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 Connections with Wallets (mTLS)

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

Perform the following steps before connecting to the database:

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

2. **For mutual TLS connections**, **Download Client Credentials**: Unzip the `wallet_databasename.zip` to a secure location. Make sure that only authorized users have access to these files.
See Download Client Credentials (Wallets) for information on downloading client credentials for Autonomous Database.

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

4. **Check JDBC Driver Version:** Download 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 and Wallets

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

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

```
dbname_high= (description=
    (address=(protocol=tcps)(port=1522)(host=adb.example.oraclecloud.com))
    (connect_data=(service_name=dbname_high.oraclecloud.com))
    (security=(ssl_server_cert_dn="CN=adb.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"
```

See [Predefined Database Service Names for Autonomous Database](#) for more details.

### Using a JDBC Connection with 18.3 JDBC Driver

Applications that use JDBC Thin driver can connect to Autonomous Databases using either Oracle Wallets or Java KeyStore (JKS).

**Using Oracle Wallet**

To use Java and the 18.3 JDBC Thin Driver to connect to Autonomous Database with the Oracle Wallet, do the following:

1. **Make sure that the prerequisites are met:** See JDBC Thin Driver Connection Prerequisites Connections with Wallets (mTLS) for more information.

2. **Verify the connection:** You can either use a Java program, a servlet, or IDEs to verify the connection to the database. A simple test is to download `DataSourceSample.java` or `UCPSample.java` from JDBC code samples and update the connection URL to have the required TNS alias and pass TNS_ADMIN, providing the path for `tnsnames.ora` and the wallet files. Also, in the sample source code update the database username and password. For example:

```
DB_URL="jdbc:oracle:thin:@dbname_high?TNS_ADMIN=/Users/test/wallet_dbname"
```
Note:
If you are using Microsoft Active Directory with a database, then in the sample source code update the username with the Active Directory username and update the password with the Active Directory user password. See Use Microsoft Active Directory with Autonomous Database for more information.

3. **Set the wallet location**: The properties file `ojdbc.properties` is pre-loaded with the wallet related connection property.

```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:

```java
java -classpath .:/lib/ojdbc8.jar:/lib/ucp.jar:/lib/oraclepki.jar:/lib/osdt_core.jar:/lib/osdt_cert.jar:. UCPSample
```

Note:
The auto-login wallet part of Autonomous Database downloaded client credentials zip file removes the need for your application to use username/password authentication.

**Using Java KeyStore**

To use Java and the 18.3 JDBC Thin Driver to connect to Autonomous Database with Java KeyStore (JKS), do the following:

1. **Make sure that the prerequisites are met**: See JDBC Thin Driver Connection Prerequisites Connections with Wallets (mTLS) for more information.

2. **Ready the database details**: You can either use a Java program, a servlet, or IDEs to check the connection to your database. A simple test is to download `DataSourceSample.java` or `UCPSample.java` from JDBC code samples. In this sample, use the connection URL as shown. Note that the connection DB_URL contains the TNS alias, for example, `dbname_high` present in `tnsnames.ora`. You can provide the path for `tnsnames.ora` file through `TNS_ADMIN` property as shown in the URL. Make sure to use the database username and password related to your database.

```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 downloading the client credentials .zip file.

To use SSL connectivity instead of Oracle Wallet, specify the keystore and truststore files and their respective password in the `ojdbc.properties` file as follows:

```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:

```bash
java -classpath ./lib/ojdbc8.jar:/lib/ucp.jar UCPSample
```

### Connecting Using JDBC Thin Driver 12.2 or Older

If you are using the JDBC driver 12.2.0.2 or older, set the Java properties prior to starting the application. Usually you set the properties in the application's startup script.

If you are not able to use the latest 18.3 JDBC drivers, then you can connect to Autonomous Database using 12.2.0.2 or other older JDBC drivers. The 12.2 or older JDBC drivers do not support the `ojdbc.properties` file. With older JDBC driver versions, you need to pass wallets or JKS related properties either as system properties or as connection properties to establish a connection.
Using Oracle Wallet

To use Java and the 12.2 or older JDBC Drivers to connect to Autonomous Database with the Oracle Wallet, do the following:

1. **Make sure that the prerequisites are met:** See JDBC Thin Driver Connection Prerequisites Connections with Wallets (mTLS) for more information.

2. **Verify the connection:** You can either use a Java program, a servlet, or IDEs to verify the connection to the database. A simple test is to download DataSourceSample.java or UCPSample.java from JDBC code samples and update the connection URL to have the required TNS alias. Also, update the sample source code to use the database username and password. For example:

   ```
   DB_URL="jdbc:oracle:thin:@dbname_high"
   ```

   **Note:**
   If you are using Microsoft Active Directory with Autonomous Database, then update the sample source code to use the Active Directory username and Active Directory user password. See Use Microsoft Active Directory with Autonomous Database for more information.

3. **Set the wallet location:** Add the OraclePKIProvider at the end of the provider list in the file java.security (this file is part of your JRE install located at $JRE_HOME/jre/lib/security/java.security) which typically looks like:

   ```
   security.provider.14=oracle.security.pki.OraclePKIProvider
   ```

4. **Compile and Run:** Compile and run the sample to get a successful connection. Make sure to have oraclepki.jar, osdt_core.jar, and osdt_cert.jar, in the classpath. Also, you need to pass the connection properties. Update the properties with the location where tnsnames.ora and wallet files are located.

   ```
   java -classpath
   ./lib/ojdbc8.jar:/lib/ucp.jar:/lib/oraepki.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
   ```

   **Note:**
   These are Windows system examples. Add a \ continuation character if you are setting -D properties on multiple lines on UNIX (Linux or a Mac).
Using Java KeyStore

To use Java and the 12.2 or older JDBC Thin Drivers to connect to Autonomous Database with Java KeyStore (JKS), do the following:

1. **Make sure that the prerequisites are met**: See JDBC Thin Driver Connection Prerequisites Connections with Wallets (mTLS) for more information.

2. **Verify the connection**: You can either use a Java program, a servlet, or IDEs to verify the connection to the database. A simple test is to download DataSourceSample.java or UCPSample.java from JDBC code samples and update the connection URL to have the required TNS alias and pass TNS_ADMIN, providing the path for tnsnames.ora and update the connection URL to have the required TNS alias. Also, in the sample source code update the database username and password. For example:

   ```
   DB_URL="jdbc:oracle:thin:@dbname_high"
   ```

   **Note:**
   If you are using Microsoft Active Directory with Autonomous Database, then update the sample source code to use the Active Directory username and Active Directory user password. See Use Microsoft Active Directory with Autonomous Database for more information.

3. **Compile and Run**: Compile and run the sample to get a successful connection. You need to pass the connection properties as shown. Update the properties with the location where tnsnames.ora and JKS files are placed. If you want to pass these connection properties programmatically then refer to DataSourceForJKS.java. For example:

   ```
   java
   -Doracle.net.tns_admin=/users/test/wallet_dbname
   -Djavax.net.ssl.trustStore=truststore.jks
   -Djavax.net.ssl.trustStorePassword=**********
   -Djavax.net.ssl.keyStore=keystore.jks
   -Djavax.net.ssl.keyStorePassword=************
   -Doracle.net.ssl_server_dn_match=true
   -Doracle.net.ssl_version=1.2 // Not required for 12.2
   ```

JDBC Thin Connections with an HTTP Proxy

If the client is behind a firewall and your network configuration requires an HTTP proxy to connect to the internet, you need to use the JDBC Thin Client 18.1 or higher which enables connections through HTTP proxies.

To connect to Autonomous Database through an HTTPS proxy, open and update your tnsnames.ora file. Add the HTTP proxy hostname(https_proxy) and port (https_proxy_port) to the connection string. Replace the values with your HTTPS proxy information. For example:

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

```sql
db2022adb_high =
  (description=
    (address=
      (https_proxy=proxyhostname)(https_proxy_port=80)
      (protocol=tcp$)(port=1522)(host=adb.example.oraclecloud.com)
    )
    (connect_data=(service_name=db2022adb_high.adb.oraclecloud.com)
    )
  
  (security=(ssl_server_cert_dn="adb.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.

**JDBC Thin TLS Connections without a Wallet**

Autonomous Database mandates a secure connection that uses Transport Layer Security (TLSv1.2). Depending on the configuration options, Autonomous Database supports mTLS and TLS authentication. This section covers using JDBC Thin Connections with TLS authentication without a wallet.

**JDBC Thin Driver Connection Prerequisites for TLS Connections without a Wallet**

Applications that use JDBC Thin driver support TLS and mutual TLS (mTLS) authentication. Connecting to an Autonomous Database instance with TLS authentication requires database
credentials (username and password) and provides a secure connection, but does not require that you download Oracle wallets or Java KeyStore (JKS) files.

---

**Note:**

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

Perform the following steps before connecting to the database:

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

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

3. **Check JDBC Driver Version**: Download the latest 18.3 JDBC Thin driver (ojdbc8.jar and ucp.jar) from Oracle Database 18c (18.3) JDBC Driver & UCP Downloads. Use the latest 18.3 JDBC driver, or newer, to take advantage of recent enhancements that simplify connections and provide easy steps for configuration. You also need the additional jars: oraclepki.jar, osdt_core.jar, and osdt_cert.jar for use with Oracle wallets.

**Using a JDBC URL TLS Connection String for JDBC Thin Driver without a Wallet**

To connect the database using JDBC Thin Driver with TLS without a wallet, you provide a connection string. Each database service has its own TNS Name and connection string.

To run an application using the JDBC Thin Driver with TLS authentication without a wallet:

1. **Copy a connection string for the Autonomous Database**.

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

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

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

2. **Set the DB_URL parameter**.

   Use the following format for the **DB_URL** parameter:

   ```
   DB_URL=jdbc:oracle:thin:@my_connect_string
   ```

---
For example:

```
DB_URL=jdbc:oracle:thin:@(description=(retry_count=20)(retry_delay=3)
(address=(protocol=tcp)
(port=1521)(host=adb.region.oraclecloud.com))
(connect_data=(service_name=u9adutfb2ba8x4d_database_medium.adb.oracleclou
d.com))
(security=(ssl_server_cert_dn="CN=adwc-preprod.uscom-east-1.oraclecloud.com, OU=Oracle BMCS US,
O=Oracle Corporation, L=Redwood City, ST=California, C=US"))
```

Preparing for Oracle Call Interface Connections

Using TLS with Oracle Call Interface connections with newer Oracle Instant Client/Oracle Database Client versions, a wallet is not required. For TLS connections with Oracle Call Interface running on older Oracle Instant Client/Oracle Database Client versions, a wallet is required for both mTLS authenticated connections and TLS authenticated connections.

Topics:

- Oracle Call Interface (OCI) Connections and Wallets (mTLS)
- Oracle Call Interface (OCI) Connections with TLS Authentication

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

Oracle Net Services can find the location of the Autonomous Database wallet using the WALLET_LOCATION parameter in the `sqlnet.ora` file.

When WALLET_LOCATION is used, Oracle Net Services automatically uses the wallet. The wallet is used transparently to the application. See Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections with Wallets (mTLS) for information on setting WALLET_LOCATION.

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

Oracle Call Interface (OCI) Connections with TLS Authentication

Using TLS with Oracle Call Interface, connections with newer Oracle Instant Client/Oracle Database Client versions, a wallet is not required.

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

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

Note:

See Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication for information on allowing TLS connections.
See Prepare for Oracle Call Interface, ODBC, and JDBC OCI Connections Using TLS Authentication to prepare for Oracle Call Interface connections.

When a wallet is required and you set WALLET_LOCATION parameter in the sqlnet.ora file, Oracle Net Services finds the location of the wallet and uses the wallet (use of the wallet is transparent to the application).

Predefined Database Service Names for Autonomous Database

The tnsnames.ora file provided with the credentials zip file contains the database service names that allow you to connect to your database.

The available predefined service names differ depending on your workload. The different services provide different levels of performance and concurrency for Autonomous Database connections.

<table>
<thead>
<tr>
<th>Workload Type</th>
<th>Database Services Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Warehouse</td>
<td><code>dbname_high</code></td>
</tr>
<tr>
<td></td>
<td><code>dbname_medium</code></td>
</tr>
<tr>
<td></td>
<td><code>dbname_low</code></td>
</tr>
<tr>
<td></td>
<td>See Database Service Names for Autonomous Data Warehouse for more information.</td>
</tr>
<tr>
<td>Transaction Processing</td>
<td><code>dbname_tpurgent</code></td>
</tr>
<tr>
<td></td>
<td><code>dbname_tp</code></td>
</tr>
<tr>
<td></td>
<td><code>dbname_high</code></td>
</tr>
<tr>
<td></td>
<td><code>dbname_medium</code></td>
</tr>
<tr>
<td></td>
<td><code>dbname_low</code></td>
</tr>
<tr>
<td></td>
<td>See Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database for more information.</td>
</tr>
<tr>
<td>JSON Database</td>
<td><code>dbname_tpurgent</code></td>
</tr>
<tr>
<td></td>
<td><code>dbname_tp</code></td>
</tr>
<tr>
<td></td>
<td><code>dbname_high</code></td>
</tr>
<tr>
<td></td>
<td><code>dbname_medium</code></td>
</tr>
<tr>
<td></td>
<td><code>dbname_low</code></td>
</tr>
<tr>
<td></td>
<td>See Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database for more information.</td>
</tr>
</tbody>
</table>

Connect with Oracle Analytics Desktop

Oracle Analytics Desktop makes it easy to visualize your data so you can focus on exploring interesting data patterns. Just connect to Autonomous Database, select the elements that you're interested in, and let Oracle Analytics Desktop find the best way to visualize it. Choose from a variety of visualizations to look at data in a specific way.

For details on connecting Autonomous Database with Oracle Analytics Desktop, see User’s Guide for Oracle Analytics Desktop.
Connect with Oracle Analytics Cloud

Oracle Analytics Cloud is a scalable and secure public cloud service that provides a full set of capabilities to explore and perform collaborative analytics for you, your workgroup, and your enterprise.

For details for connecting Autonomous Database with Oracle Analytics Cloud, see Visualizing Data and Building Reports in Oracle Analytics Cloud.

Connection and Networking Options and Features

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

Topics

• Using ACLs, VCNs, and Private Endpoints with Autonomous Database
• Connect with Oracle Cloud Infrastructure FastConnect
• Access Autonomous Database with VCN Transit Routing
• Access Autonomous Database with Service Gateway
• 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 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 database with the **Allow secure access from everywhere** option, you can add, modify, or remove ACLs after you provision or clone the database.

See **Overview of Restricting Access with ACLs** for more information.

• **Virtual cloud network**: This option assigns a private endpoint, private IP, and hostname to your database. Specifying this option allows traffic only from the VCN you specify; access to the database from all public IPs or VCNs is blocked. This allows you to define security rules, ingress/egress, at the Network Security Group (NSG) level and to control traffic to your Autonomous Database.

See **Configure Private Endpoints When You Provision or Clone an Instance** for more information.

See **Overview of Restricting Access with ACLs**

When you select the network access **Allow secure access from anywhere** option when you provision or clone an instance, you can restrict network access by defining an Access Control List (ACL). You can also add, update, or remove an ACL for an active instance.
Specifying an access control list blocks all IP addresses that are not in the ACL list from accessing the database. After you specify an access control list, the database only accepts connections from addresses on the access control list and the database rejects all other client connections.

Depending on where the client machines that connect to your database are located you have the following options with ACLs:

- If your client machines connect to your database through the public internet, then you can use ACLs to specify the client machine's public IP addresses or their public CIDR blocks. In this case only the specified public IP addresses can access your database.

- If the client machines reside in an Oracle Cloud Infrastructure Virtual Cloud Network (VCN), you can configure a Service Gateway to connect to your database. In this case, you can specify the VCN in your ACL, this allows all client machines in that VCN to access your database and blocks all other connections. Furthermore, you can specify the VCN and a list of private IP addresses or CIDR blocks in that VCN. This allows only those client machines with the specified IP addresses or CIDR blocks to access your database and blocks all other connections.

See VCNs and Subnets for details on Virtual Cloud Networks (VCN).

See Access to Oracle Services: Service Gateway for details on setting up a Service Gateway.

- If you have on-premises clients that connect to your database through Transit Routing, you can specify the VCN and also the private IP addresses or CIDR blocks of these on-premises clients to access to your database.


- You can use these options together to set multiple rules to allow access from different types of clients. Multiple rules do not exclude each other.

See Configuring Network Access with Access Control Rules (ACLs) for the steps for configuring network access with ACLs, either when you provision or clone your database, or whenever you want to add, modify or remove ACLs.

Overview of Private Endpoints

You can specify that Autonomous Database uses a private endpoint inside your Virtual Cloud Network (VCN) in your tenancy. You can configure a private endpoint during provisioning or cloning your Autonomous Database, or you can switch to using private endpoints in existing databases that use public endpoints. This allows you to keep all traffic to and from your Autonomous Database off of the public internet.

Specifying the Virtual cloud network configuration option only allows traffic from the VCN you specify and blocks access to the database from all public IPs or VCNs. This allows you to define security rules, ingress/egress, at the Network Security Group (NSG) level and to control traffic to your database.

See Configuring Network Access with Private Endpoints for the steps for configuring network access private endpoints, either when you provision or clone your database, or whenever you want to add, modify or remove private endpoints.

Connect with Oracle Cloud Infrastructure FastConnect

Oracle Cloud Infrastructure FastConnect provides an easy way for you to connect your on-premises network to Autonomous Database using FastConnect Public Peering. FastConnect
provides higher-bandwidth options, and a more reliable and consistent networking experience compared to internet-based connections.

Use FastConnect to access public services in Oracle Cloud Infrastructure without using the internet, for example, access to Object Storage, or the Oracle Cloud Infrastructure Console and APIs. Without FastConnect, the traffic destined for public IP addresses would be routed over the internet. With FastConnect, that traffic goes over your private physical connection.

For details for connecting Autonomous Database with Oracle Cloud Infrastructure FastConnect see FastConnect Overview.

Access Autonomous Database with VCN Transit Routing

Oracle Cloud Infrastructure supports private access from your on-premises network to a database with virtual cloud network (VCN) Transit Routing.

Transit routing refers to a network setup in which your on-premises network uses a connected virtual cloud network (VCN), through a service gateway on the VCN for connectivity to a database. You connect the on-premises network to the VCN with FastConnect or VPN Connect, and then configure the VCN routing so that traffic transits through the VCN to a database beyond the VCN.

After transit routing is configured, there are no special steps required to access Oracle APEX or Database Actions, or to consume RESTful services published in Oracle REST Data Services.

See Transit Routing: Private Access to Oracle Services for more details and options available with Transit Routing.

See VPN Connect for more information on VPN Connect.

Access Autonomous Database with Service Gateway

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

A service gateway allows connectivity to Autonomous Database from private IP addresses in private subnets without requiring a NAT Gateway in your VCN. After a service gateway is configured, there are no special steps required to connect to Autonomous Database. Use the same connection steps as described in this chapter, depending on your application type or the client tool you are using to connect to the database.

After a service gateway is configured, there are no special steps required to access Oracle APEX or Database Actions, or to consume RESTful services published in Oracle REST Data Services.

See Access to Oracle Services: Service Gateway for details.

Use Database Resident Connection Pooling with Autonomous Database

Database Resident Connection Pool (DRCP) in Autonomous Database supports easier and more efficient management of open connections. Using DRCP provides you with access to a connection pool in your database that enables a significant
reduction in key database resources required to support many client connections and when the database needs to scale for many simultaneous connections.

When you connect to Autonomous Database you choose one of the following depending on values specified in the tnsnames.ora configuration file:

- A dedicated server process, which services only one user process.
- A pooled server process, obtained from DRCP, which can service multiple user processes.

To connect with a pooled DRCP server process, do the following:

1. Locate or obtain the tnsnames.ora file you are using to connect to your Autonomous Database.
   See Download Client Credentials (Wallets) for more information.
2. Modify the tnsnames.ora file to add the server type SERVER=POOLED.
   For example:

   ```
   example_high = (description=
   (address=(protocol=tcps)(port=1522)
   (host=adb.example.oraclecloud.com))
   (connect_data=(service_name=example_high.oraclecloud.com)
   (SERVER=POOLED))
   (security=(ssl_server_cert_dn="CN=adb.oraclecloud.com,OU=Oracle US,0=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.
   See Using Database Resident Connection Pool for more information.

Use Database Links with Autonomous Database

You can create database links to other Oracle databases or to non-Oracle databases. In addition, you can create database links from other databases to Autonomous Databases.

Topics

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

You can create database links from Autonomous Database to another publicly accessible Oracle Database or to an Oracle database that is on a private endpoint.

Topics

- Create Database Links from Autonomous Database to Publicly Accessible Oracle Databases
- Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint
- Database Link Notes

Create Database Links from Autonomous Database to Publicly Accessible Oracle Databases

Use `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to create database links from an Autonomous Database to an Oracle database that is publicly accessible.

To create database links to a target on a private endpoint, see Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint.

To use database links with Autonomous Database the target database must be configured to use TCP/IP with SSL (TCPS) authentication. Autonomous Databases use TCP/IP with SSL (TCPS) authentication by default, so you do not need to do any additional configuration in your target database to link to another Autonomous Database. Other Oracle databases must be configured to use TCP/IP with SSL (TCPS) authentication. See Configuring Secure Sockets Layer Authentication for more information.

To ensure security, the database link port is restricted to the range 1521-1525, or port 2484. You specify the target database port when you create a database link with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

To create database links to a public target, the target database must be accessible. Some databases, including Autonomous Databases, may limit access (for example, using Access Control Lists). Make sure you enable your target database to allow access from your source database for the database link to work. If you limit access with Access Control Lists (ACLs), you can find the outbound IP address of your source Autonomous Database and allow that IP address to connect to your target database. For example, if the target database is another Autonomous Database, you can add the outbound IP address of the source database to the ACL of the target database. See Obtain Tenancy Details for information on finding the outbound IP address.

To create database links to a target Oracle 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.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Create a directory on Autonomous Database for the wallet file `cwallet.sso`.

For example:

```
CREATE DIRECTORY dblink_wallet_dir AS 'directory_path_of_your_choice';
```

See Create Directory in Autonomous Database for information on creating directories.

4. Use DBMS_CLOUD.GET_OBJECT to upload the target database wallet to the directory you created in the previous step, `DBLINK_WALLET_DIR`.

For example:

```
BEGIN
    DBMS_CLOUD.GET_OBJECT(
        credential_name => 'DEF_CRED_NAME',
        object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/cwallet.sso',
        directory_name => 'DBLINK_WALLET_DIR');
END;
/
```

In this example, `namespace-string` is the Oracle Cloud Infrastructure object storage namespace and `bucketname` is the bucket name. See Understanding Object Storage Namespaces for more information.

Note:

The credential_name you use in this step is the credentials for the Object Store. In the next step you create the credentials to access the target database.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

5. On Autonomous Database create credentials to access the target database. The username and password you specify with DBMS_CLOUD.CREATE_CREDENTIAL are the credentials for the target database that you use to create the database link.
Note:
Supplying the `credential_name` parameter is required.

For example:

BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'DB_LINK_CRED',
        username => 'NICK',
        password => 'password'
    );
END;
/

The characters in the `username` parameter must be all uppercase letters.

This operation stores the credentials in the database in an encrypted format. You can use any name for the `credential_name`.

6. Create the database link to the target database using
   `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

For example:

```
BEGIN
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'SALESLINK',
        hostname => 'adb.eu-frankfurt-1.oraclecloud.com',
        port => '1522',
        service_name => 'example_medium.adb.example.oraclecloud.com',
        ssl_server_cert_dn => 'CN=adb.example.oraclecloud.com,OU=Oracle BMCS FRANKFURT,O=Oracle Corporation,L=Redwood City,ST=California,C=US',
        credential_name => 'DB_LINK_CRED',
        directory_name => 'DBLINK_WALLET_DIR');
END;
/
```

Users other than `ADMIN` require privileges to run
`DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

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

For example:

```
SELECT * FROM employees@SALESLINK;
```
For the credentials you create in Step 5, the target database credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:

```
BEGIN
    DBMS_CLOUD.UPDATE_CREDENTIAL (  
        credential_name => 'DB_LINK_CRED',  
        attribute => 'PASSWORD',  
        value => 'password');
END;
/
```

Where `password` is the new password.

After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

**Note:**

You can create links to Big Data Service using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. See [Query Big Data Service Hadoop (HDFS) Data from Autonomous Database](#) for more information.

For additional information, see:

- [CREATE_DATABASE_LINK Procedure](#)
- [GET_OBJECT Procedure and Function](#)
- [UPDATE_CREDENTIAL Procedure](#)
Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint

You can create database links from an Autonomous Database to a target Oracle database that is on a private endpoint.

**Note:**

Database links from an Autonomous Database to a target Oracle database that is on a private endpoint are only supported in commercial regions and US Government regions. This feature is enabled by default in all commercial regions.

This feature is enabled by default in US Government regions for newly provisioned databases.

For existing US Government databases on a private endpoint, if you want to create database links from an Autonomous Database to a target in a US Government region, you can file a Service Request at Oracle Cloud Support and request to enable the private endpoint in government regions database linking feature.

US Government regions include the following:

- Oracle Cloud Infrastructure US Government Cloud with FedRAMP Authorization
- Oracle Cloud Infrastructure US Federal Cloud with DISA Impact Level 5 Authorization

Depending on the type and the configuration of the target Oracle database:

- **Autonomous Database or another Oracle Database, such as on-premises or a Database Cloud Service database, on a private endpoint that is configured for SSL (TCPS):** In this case you create the database link with a wallet and the database link communicates with TCPS. See Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint with a Wallet for details.

- **Oracle Database, such as on-premises or a Database Cloud Service database, on a private endpoint that is configured for TCP:** In this case you create the database link without a wallet and the database link communicates with TCP. See Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint without a Wallet for details.

**Topics**

- Prerequisites for Database Links from Autonomous Database to Oracle Databases on a Private Endpoint
- Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint with a Wallet
Prerequisites for Database Links from Autonomous Database to Oracle Databases on a Private Endpoint

Lists the prerequisites to create database links from an Autonomous Database to a target Oracle database that is on a private endpoint.

To create a database link to a target Oracle database on a private endpoint:

- The target database must be accessible from the source database's Oracle Cloud Infrastructure VCN. For example, you can connect to the target database when:
  - The target database is on a private endpoint.
  - Both the source database and the target database are in the same Oracle Cloud Infrastructure VCN.
  - The source database and the target database are in different Oracle Cloud Infrastructure VCNs that are paired.
  - The target database is an on-premises database that is connected to the source database's Oracle Cloud Infrastructure VCN using FastConnect or VPN.

- The following ingress and egress rules must be defined for the private endpoint:
  - Define an egress rule in the source database's subnet security list or network security group such that the traffic over TCP is allowed to the target database's IP address and port number.
  - Define an ingress rule in the target database's subnet security list or network security group such that the traffic over TCP is allowed from the source database IP address to the destination port.

See Configuring Network Access with Private Endpoints for information on configuring private endpoints with ingress and egress rules.

- To ensure security, the database link port is restricted to the range 1521-1525, or port 2484. You specify the target database port when you create a database link with DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK.

**Note:**

When your Autonomous Database instance is configured with a private endpoint, set the ROUTE_OUTBOUND_CONNECTIONS database parameter to 'PRIVATE_ENDPOINT' to specify that all outgoing database links are subject to the Autonomous Database instance private endpoint VCN's egress rules. See Enhanced Security for Outbound Connections with Private Endpoints for more information.
Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint with a Wallet

Use `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to create database links from an Autonomous Database to a target Oracle database that is on a private endpoint.

**Note:**

This option is for target Oracle databases that have SSL/TCPS configured and that are on a private endpoint. Autonomous Database has TCP/IP with SSL (TCPS) authentication configured by default, so you can use these steps to connect to another Autonomous Database.

If the target Oracle database does not have SSL/TCPS configured, you have two options:

- You can configure the target Oracle database to use TCP/IP with SSL (TCPS) authentication. See Configuring Transport Layer Security Authentication for information on configuring SSL/TCPS.
- You can connect to the target Oracle database with TCP. See Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint without a Wallet for details.

Perform the prerequisite steps, as required. See Prerequisites for Database Links from Autonomous Database to Oracle Databases on a Private Endpoint for details.

To create a database link to a target database on a private endpoint using TCP/IP with SSL (TCPS) authentication:

1. Copy your target database wallet, `cwallet.sso`, containing the certificates for the target database to Object Store.

   **Note:**

   The wallet file, along with the Database user ID and password provide access to data in the target Oracle database. Store wallet files in a secure location. Share wallet files only with authorized users.

2. Create credentials to access your Object Store where you store the `cwallet.sso`. See `CREATE_CREDENTIAL Procedure` for information about the username and password parameters for different object storage services.

3. Create a directory on Autonomous Database for the wallet file `cwallet.sso`. For example:

   ```sql
   CREATE DIRECTORY wallet_dir AS 'directory_path_of_your_choice';
   ```

   See Create Directory in Autonomous Database for information on creating directories.
4. Use `DBMS_CLOUD.GET_OBJECT` to upload the target database wallet to the directory you created in the previous step, `WALLET_DIR`.

For example:

```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 => 'WALLET_DIR');
END;
/
```

In this example, `namespace-string` is the Oracle Cloud Infrastructure object storage namespace and `bucketname` is the bucket name. See Understanding Object Storage Namespaces for more information.

**Note:**

The `credential_name` you use in this step is the credentials for the Object Store. In the next step you create the credentials to access the target database.

5. On Autonomous Database create credentials to access the target database. The `username` and `password` you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database used within the database link, (where the target database is accessed through the VCN).

**Note:**

Supplying the `credential_name` parameter is required.

For example:

```sql
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'DB_LINK_CRED',
        username => 'NICK',
        password => 'password');
END;
/
```

The characters in the `username` parameter must be all uppercase letters.

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

6. Create the database link to the target database using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. 
For example:

```
BEGIN
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'PEDBLINK1',
        hostname => 'example1.adb.ap-osaka-1.oraclecloud.com',
        port => '1522',
        service_name => 'example_high.adb.oraclecloud.com',
        ssl_server_cert_dn => 'ssl_server_cert_dn',
        credential_name => 'DB_LINK_CRED',
        directory_name => 'WALLET_DIR',
        private_target => TRUE);
END;
/
```

**Note:**

If you set `ROUTE_OUTBOUND_CONNECTIONS` to `PRIVATE_ENDPOINT`, setting the `private_target` parameter to `TRUE` is not required in this API. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

```
DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK only supports a single hostname for the hostname parameter. Using an IP address or SCAN IP is not supported.

Users other than ADMIN require privileges to run `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

7. Use the database link you created to access data in the target database.
   For example:

   ```
   SELECT * FROM employees@PEDBLINK1;
   ```
**Note:**

For the credentials you create in Step 5, the Oracle Database credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:

```
BEGIN
    DBMS_CLOUD.UPDATE_CREDENTIAL (credential_name => 'DB_LINK_CRED',
                                    attribute => 'PASSWORD',
                                    value => 'password');
END;
/
```

Where `password` is the new password.

After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

See [CREATE_DATABASE_LINK Procedure](#) for additional information.

### Create Database Links from Autonomous Database to Oracle Databases on a Private Endpoint without a Wallet

Use `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to create database links from an Autonomous Database to a target Oracle database that is on a private endpoint and connect without a wallet (TCP).

**Note:**

This option is for target Oracle databases that are on a private endpoint and do not have SSL/TCPS configured.

Perform the prerequisite steps, as required. See [Prerequisites for Database Links from Autonomous Database to Oracle Databases on a Private Endpoint](#) for details.

To create a database link to a target database on a private endpoint using a secure TCP connection without a wallet:

1. On Autonomous Database create credentials to access the target database. The `username` and `password` you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database used within the database link, (where the target database is accessed through the VCN).

   For example:

   ```
   BEGIN
       DBMS_CLOUD.CREATE_CREDENTIAL(
           credential_name => 'PRIVATE_ENDPOINT_CRED',
           username => 'NICK',
           password => 'password'
   ```
The characters in the username parameter must be all uppercase letters.

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

2. Create the database link to the target database using DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK.

For example:

BEGIN
   DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(  
      db_link_name => 'PRIVATE_ENDPOINT_LINK',
      hostname => 'exampleHostname',
      port => '1522',
      service_name => 'exampleServiceName',
      ssl_server_cert_dn => NULL,
      credential_name => 'PRIVATE_ENDPOINT_CRED',
      directory_name => NULL,
      private_target => TRUE);  
END;
/

DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK only supports a single hostname for the hostname parameter. Using an IP address or SCAN IP is not supported.

Users other than ADMIN require privileges to run DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK.

As shown in the example, to create a database link with DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK to a target database on a private endpoint using a secure TCP connection without a wallet, all of the following are required:

- The directory_name parameter must be NULL.
- The ssl_server_cert_dn parameter must be NULL.
- The private_target parameter must be TRUE.

Note:

If you set ROUTE_OUTBOUND_CONNECTIONS to PRIVATE_ENDPOINT, setting the private_target parameter to TRUE is not required in this API. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

- To ensure security, the database link port you specify in the port parameter is restricted to the range 1521-1525, or port 2484.

3. Use the database link you created to access data in the target database.
For example:

```sql
SELECT * FROM employees@PRIVATE_ENDPOINT_LINK;
```

**Note:**

For the credentials you create in Step 1, the Oracle Database credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:

```sql
BEGIN
  DBMS_CLOUD.UPDATE_CREDENTIAL (
    credential_name => 'DB_LINK_CRED',
    attribute => 'PASSWORD',
    value => 'password');
END;
/
```

Where `password` is the new password.

After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

See [CREATE_DATABASE_LINK Procedure](#) for additional information.

### Database Link Notes

Notes for database links to other Databases:

- Only one wallet file is valid per directory for use with database links. You can only upload one `cwallet.sso` at a time to the directory you choose for wallet files (for example `DBLINK_WALLET_DIR`). This means with a `cwallet.sso` in `DBLINK_WALLET_DIR` you can only create database links to the databases for which the wallet in that directory is valid. To use multiple `cwallet.sso` files with database links you need to create additional directories and put each `cwallet.sso` in a different directory. When you create database links with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`, specify the directory that contains the wallet with the `directory_name` parameter.

  See [Create Directory in Autonomous Database](#) for information on creating directories.

- Supported target Oracle database versions for database links to another Oracle Database are: 19c, 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](#).

- 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.

Create Database Links to Non-Oracle Databases

You can create database links from Autonomous Database to Non-Oracle databases.

Topics

- Create Database Links to Non-Oracle Databases with Oracle-Managed Heterogeneous Connectivity
- Create Database Links to Non-Oracle Databases with Customer-Managed Heterogeneous Connectivity

Create Database Links to Non-Oracle Databases with Oracle-Managed Heterogeneous Connectivity

Autonomous Database support for Oracle-managed heterogeneous connectivity makes it easy to create database links to non-Oracle databases. When you use database links with Oracle-managed heterogeneous connectivity, Autonomous Database configures and sets up the connection to the non-Oracle database.

The following are prerequisites to use Oracle-managed heterogeneous connectivity with Autonomous Database:

- The target database must be accessible from the public internet on the port number supported for the specified database type. See Oracle-Managed Heterogeneous Connectivity Database Types and Ports for the list of supported non-Oracle database types and ports.
- The target database must be configured to allow incoming SSL/TLS connections.

Note:

Oracle-managed heterogeneous connectivity supports connections to Oracle MySQL Database Service on private endpoints. When you connect to MySQL on a private endpoint, the connections uses TCP protocol and it does not require SSL/TLS to be configured on the target database. See Create Database Links to Oracle MySQL on a Private Endpoint with Oracle-Managed Heterogeneous Connectivity for more information.

To create database links to a non-Oracle database using Oracle-managed heterogeneous connectivity, do the following:

1. On Autonomous Database create credentials to access the target database. The username and password you specify with DBMS_CLOUD.CREATE_CREDENTIAL are the credentials for the target database used within the database link.

Note:

Supplying the credential_name parameter is required.
For example:

BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'AWS_REDSHIFT_LINK_CRED',
        username => 'nick',
        password => 'password'
    );
END;
/

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

2. Create the database link to the target database using DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK.

For example, to create a database link to AWS Redshift:

BEGIN
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'AWSREDSHIFT_LINK',
        hostname => 'example.com',
        port => '5439',
        service_name => 'example_service_name',
        credential_name => 'AWS_REDSHIFT_LINK_CRED',
        gateway_params => JSON_OBJECT('db_type' value 'AWSREDSHIFT'),
        ssl_server_cert_dn => NULL);
END;
/

The hostname for Snowflake is your account identifier. To find your Snowflake account identifier, see Account Identifier Formats by Cloud Platform and Region.

The service_name is the database name of the non-Oracle database.

The gateway_params db_type value that you supply must be one of the supported values:

<table>
<thead>
<tr>
<th>db_type Value</th>
<th>Database Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWSREDSHIFT</td>
<td>Amazon Redshift</td>
</tr>
<tr>
<td>AZURE</td>
<td>Microsoft SQL Server</td>
</tr>
<tr>
<td></td>
<td>Azure SQL</td>
</tr>
<tr>
<td></td>
<td>Azure Synapse Analytics</td>
</tr>
<tr>
<td>MYSQL</td>
<td>MySQL</td>
</tr>
<tr>
<td>POSTGRES</td>
<td>PostgreSQL</td>
</tr>
<tr>
<td>SNOWFLAKE</td>
<td>Snowflake</td>
</tr>
</tbody>
</table>

Autonomous Database automatically configures and handles the secure connection to a target database and your connections are end-to-end encrypted. Oracle-managed heterogeneous connectivity is preconfigured with a wallet that contains most of the common trusted root and intermediate SSL certificates. Thus, NULL must be provided as the value for the ssl_server_cert_dn parameter.
To ensure security when using database links with Oracle-managed heterogeneous connectivity, the connection port is restricted and must have SSL/TLS enabled. You specify the target database port with the `port` parameter.

See Oracle-Managed Heterogeneous Connectivity Database Types and Ports for the list of supported non-Oracle database types.

3. Use the database link to access data on the target database.

For example:

```sql
SELECT count(*) FROM sales@AWSREDSHIFT_LINK
```

Note:

For the credentials you create in Step 1, the target database credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:

```sql
BEGIN
    DBMS_CLOUD.UPDATE_CREDENTIAL ( credential_name => 'AWS_REDSHIFT_LINK_CRED', attribute => 'PASSWORD', value => 'password');
END;
/ 
```

Where `password` is the new password.

After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.

For additional information, see:

- Supported SQL Syntax and Functions in Oracle Database Gateway for ODBC User's Guide
- CREATE_DATABASE_LINK Procedure
- UPDATE_CREDENTIAL Procedure

Create Database Links to Oracle MySQL on a Private Endpoint with Oracle-Managed Heterogeneous Connectivity

Autonomous Database support for Oracle-managed heterogeneous connectivity makes it easy to create database links to Oracle MySQL Database Service on a private endpoint. When you use database links with Oracle-managed heterogeneous
connectivity, Autonomous Database configures and sets up the connection to the Oracle MySQL Database Service.

**Note:**

Database links from an Autonomous Database to an Oracle MySQL Database Service that is on a private endpoint are only supported in commercial regions and US Government regions. This feature is enabled by default in all commercial regions. This feature is enabled by default in US Government regions for newly provisioned databases.

For existing US Government databases on a private endpoint, if you want to create database links from an Autonomous Database to a target in a US Government region, please file a Service Request at Oracle Cloud Support and request to enable the private endpoint in government regions database linking feature.

US Government regions include the following:

- Oracle Cloud Infrastructure US Government Cloud with FedRAMP Authorization
- Oracle Cloud Infrastructure US Federal Cloud with DISA Impact Level 5 Authorization

The following are prerequisites to use Oracle-managed heterogeneous connectivity with Oracle MySQL Database Service on a private endpoint:

- Create a DNS name using private DNS Zone pointing to private IP of your Oracle MySQL Database Service in your VCN. See Private DNS.
- Create an Autonomous Database with a Private Endpoint on same subnet.
- Configure the VCN for the Oracle MySQL Database Service to allow incoming connections on port 3306.

To create database links to a Oracle MySQL Database Service on a private endpoint using Oracle-managed heterogeneous connectivity, do the following:

1. On Autonomous Database create credentials to access the Oracle MySQL Database Service. The **username** and **password** you specify with DBMS_CLOUD.CREATE_CREDENTIAL are the credentials for the Oracle MySQL Database Service used within the database link.

   **Note:**

   Supplying the **credential_name** parameter is required.

For example:

BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
      credential_name => 'MYSQL_LINK_CRED',
   )
END
username => 'NICK',
password => 'password'
);
END;
/

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

2. Create the database link to the Oracle MySQL Database Service using DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK.

For example, to create a database link:

BEGIN
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'MYSQL_LINK',
        hostname => 'mysql.example.com',
        port => '3306',
        service_name => 'mysql.example_service_name',
        ssl_server_cert_dn => NULL,
        credential_name => 'MYSQL_LINK_CRED',
        private_target => TRUE,
        gateway_params => JSON_OBJECT('db_type' value 'MYSQL'));
END;
/

3. Use the database link to access data on the target database.

For example:

SELECT count(*) FROM sales@MYSQL_LINK

---

**Note:**

For the credentials you create in Step 1, the target database credentials, if the password of the target user changes you can update the credential that contains the target user's credentials as follows:

BEGIN
    DBMS_CLOUD.UPDATE_CREDENTIAL (
        credential_name => 'MYSQL_LINK_CRED',
        attribute => 'PASSWORD',
        value => 'password');
END;
/

Where **password** is the new password.

After this operation, the existing database links that use this credential continue to work without having to drop and recreate the database links.
For additional information, see:

- Supported SQL Syntax and Functions in *Oracle Database Gateway for ODBC User's Guide*
- **CREATE_DATABASE_LINK Procedure**
- **UPDATE_CREDENTIAL Procedure**

Oracle-Managed Heterogeneous Connectivity Database Types and Ports

Shows the non-Oracle databases that you can connect to from Autonomous Database with Oracle-managed heterogeneous connectivity, and lists the supported port value for each database type. Also provides a link where you can see the supported database versions for each database type.

Note:

Oracle uses Progress DataDirect connectors. The Database Support column provides links to the Progress website where you can find the supported database versions for each database type.

<table>
<thead>
<tr>
<th>Database Type</th>
<th>Required Port</th>
<th>Database Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Redshift</td>
<td>5439</td>
<td>Amazon Redshift Supported Versions</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>1433</td>
<td>Azure SQL Supported Versions</td>
</tr>
<tr>
<td>Azure Synapse Analytics</td>
<td></td>
<td>Azure Synapse Analytics Supported Versions</td>
</tr>
<tr>
<td>MySQL</td>
<td>3306</td>
<td>MySQL Supported Versions</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>5432</td>
<td>PostgreSQL Supported Versions</td>
</tr>
<tr>
<td>Snowflake</td>
<td>443</td>
<td>Snowflake Supported Versions</td>
</tr>
</tbody>
</table>

Create Database Links to Non-Oracle Databases with Customer-Managed Heterogeneous Connectivity

Use `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to create database links from an Autonomous Database to an Oracle Database Gateway to access Non-Oracle databases.

An Oracle Database Gateway is a gateway that is designed for accessing a specific non-Oracle system. Using an Oracle Database Gateway, you can access data anywhere in a distributed database system without knowing either the location of the data or how it is stored. Using database links on Autonomous Database with Oracle Database Gateway supports heterogeneous environments and eliminates the need to customize your applications to access data from non-Oracle systems.

To use database links with Autonomous Database, the target gateway must be configured to use TCP/IP with SSL (TCPS) authentication. See Configuring Secure Sockets Layer Authentication for more information.

To ensure security when using database links with an Oracle Database Gateway, the database link port is restricted to the range 1521-1525, or port 2484. You specify the target gateway port when you create a database link with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.
Before you create database links to a target gateway, do the following:

1. Configure the Oracle Database Gateway to access a non-Oracle database.
2. Configure Oracle Net Listener to handle incoming requests on the Oracle Database Gateway.
3. Create a self signed wallet on the Oracle Database Gateway.

For more information on Oracle Database Gateway, see Oracle Database Gateways in Oracle Database Heterogeneous Connectivity User's Guide. Also see the Installation and Configuration Guide and the Gateway User's Guide for the database you want to connect to. For example, for Oracle Database Gateway for SQL Server see:

- Installing and Configuring Oracle Database Gateway for SQL Server
- Introduction to the Oracle Database Gateway for SQL Server in Oracle Database Gateway for SQL Server User's Guide.
- Configure Oracle Net for the Gateway

To create database links to a target gateway, do the following:

1. Copy the target gateway self signed wallet, for example, cwallet.sso, containing the certificates for the Oracle Database Gateway to Object Store.

   **Note:**
   
   The wallet file, along with the Database user ID and password provide access to data available through the target gateway. Store wallet files in a secure location. Share wallet files only with authorized users.

2. Create credentials to access the Object Store where you store the cwallet.sso.
   See CREATE_CREDENTIAL Procedure for information about the username and password parameters for different object storage services.

3. Create a directory on Autonomous Database for the wallet file cwallet.sso.
   For example:
   ```sql
   CREATE DIRECTORY dblink_wallet_dir AS 'directory_path_of_your_choice';
   ```
   
   See Create Directory in Autonomous Database for information on creating directories.

4. Use DBMS_CLOUD.GET_OBJECT to upload the target gateway self signed wallet to the directory you created in the previous step, DBLINK_WALLET_DIR.
   For example:
   ```sql
   BEGIN
     DBMS_CLOUD.GET_OBJECT(
   ```
In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

Note: The credential_name you use in this step is the credentials for the Object Store. In the next step you create the credentials to access the target gateway.

5. On Autonomous Database create credentials to access the target database. The username and password you specify with DBMS_CLOUD.CREATE_CREDENTIAL are the credentials for the target database used within the database link, (where the target database is accessed through the Oracle Database Gateway).

Note: Supplying the credential_name parameter is required.

For example:

BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'DB_LINK_CRED',
        username => 'NICK',
        password => 'password'
    );
END;
/

The characters in the username parameter must be all uppercase letters.

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name.

6. Create the database link to the target gateway using DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK.

For example:

BEGIN
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'SALESLINK',
        hostname => 'example.com',
        port => '1522',
    );
END;
/
service_name => 'example_service_name',
ssl_server_cert_dn => 'ssl_server_cert_dn',
credential_name => 'DB_LINK_CRED',
directory_name => 'DBLINK_WALLET_DIR',
gateway_link => TRUE);
END;
/

Users other than ADMIN require privileges to run
DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK.

7. Use the database link you created to access data on the target gateway.
   For example:

   SELECT * FROM employees@SALESLINK;

   **Note:**
   For the credentials you create in Step 5, the Oracle Database Gateway
   credentials, if the password of the target user changes you can update the
   credential that contains the target user's credentials as follows:

   BEGIN
   DBMS_CLOUD.UPDATE_CREDENTIAL (
      credential_name => 'DB_LINK_CRED',
      attribute => 'PASSWORD',
      value => 'password');
   END;
   /

   Where **password** is the new password.
   After this operation, the existing database links that use this credential
   continue to work without having to drop and recreate the database links.

See the following for an example that shows you how to create a database link to an
Oracle Database Gateway to access a Microsoft SQL Server database:

How to Access Non-Oracle Databases from Autonomous Database using Oracle
Database Gateway

For additional information, see:

- CREATE_DATABASE_LINK Procedure
- GET_OBJECT Procedure and Function
- UPDATE_CREDENTIAL Procedure
Create Database Links from Other Databases to Autonomous Database

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

To create database links to a database do the following:

1. Download your Autonomous Database wallet. See Download Client Credentials (Wallets) for more information.
2. Upload the wallet to the database instance where you want to create the link to the Autonomous Database.
3. Unzip the Autonomous Database wallet:

   Note:
   The wallet file, along with the Database user ID and password provide access to data in your Autonomous Database. Store wallet files in a secure location. Share wallet files only with authorized users.

   [oracle@sys1 ~]$ cd/u01/targetwallet
   [oracle@sys1 targetwallet]$ unzip Wallet_name1.zip
   Archive: Wallet_name1.zip
   inflating: cwallet.sso
   inflating: tnsnames.ora
   inflating: truststore.jks
   inflating: ojdbc.properties
   inflating: sqlnet.ora
   inflating: ewallet.p12
   inflating: keystore.jks

4. Set GLOBAL_NAMES to FALSE.

   SQL> ALTER SYSTEM SET GLOBAL_NAMES = FALSE;
   System altered.

   SQL> SHOW PARAMETER GLOBAL_NAMES
   NAME     TYPE  VALUE
   -----------------  --------  
   global_names     boolean  FALSE

   Set GLOBAL_NAMES to FALSE to use a database link name without checking that the name is different than the remote database name. When GLOBAL_NAMES is set to TRUE, the database requires the database link to have the same name as the database to which it connects. See GLOBAL_NAMES for more information.
5. Create the database link to the target Autonomous Database. Note that the security path includes \texttt{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.adb.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,0=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 \texttt{ALL\_DB\_LINKS} view. See \texttt{ALL\_DB\_LINKS} for more information.

   For additional information, see:

   - See \texttt{CREATE DATABASE LINK} for details on the procedure.
   - See \texttt{Create Database Links from Autonomous Database to Publicly Accessible Oracle 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 \texttt{DBMS\_CLOUD\_ADMIN.DROP\_DATABASE\_LINK}.

  For example:

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

  See \texttt{DROP\_DATABASE\_LINK Procedure} for detailed information about the procedure.
Loading Data with Autonomous Database

Describes packages and tools to load data with Autonomous Database.

Topics

• About Data Loading
• Load Data with Oracle Database Actions
• Load Data from Files in the Cloud
• Import Data Using Oracle Data Pump on Autonomous Database
• Load Data from Local Files with Oracle Database Actions
• Use Oracle GoldenGate to Replicate Data to Autonomous Database
• Load Data from Local Files Using SQL*Loader

About Data Loading

Autonomous Database provides the following loading options:

• You can load data using Oracle Database Actions.
• You can load data using Oracle Database tools and Oracle or other 3rd party data integration tools.
• On transaction processing systems you traditionally ingest data through routine transactions or with DML operations.

In general you load data from files local to your client computer or from files stored in a cloud-based object store. To load data from files in the cloud, use either Oracle Database Actions or use the Autonomous Database PL/SQL package DBMS_CLOUD to load files from the cloud.

For the fastest data loading experience Oracle recommends uploading the source files to a cloud-based object store, such as Oracle Cloud Infrastructure Object Storage, before loading the data into your database. Oracle provides support for loading files that are located locally in your data center, but when using this method of data loading you should factor in the transmission speeds across the Internet which may be significantly slower.

For more information on Oracle Cloud Infrastructure Object Storage, see Putting Data into Object Storage and Overview of Object Storage.

Note:

If you are not using ADMIN user, ensure the user has the necessary privileges for the operations the user needs to perform. See Manage User Privileges on Autonomous Database - Connecting with a Client Tool for more information.
Load Data with Oracle Database Actions

Oracle Database Actions provides a web-based interface with development tools, data tools, administration, and monitoring features and lets you load or access data from local files, from cloud storage, or from remote databases.

On the Database Actions Data Load page, you can choose to load data from a file on your local device, from cloud storage, or from a database. You can also choose to explore the data in your Oracle Autonomous Database. See The Data Load Page for detailed information and the steps for loading data using Database Actions.

See Connect with Built-in Oracle Database Actions for information on accessing Oracle Database Actions.
Load Data from Files in the Cloud

The PL/SQL package `DBMS_CLOUD` provides support for loading data from text, ORC, Parquet, and Avro files in the Cloud to your tables in Autonomous Database. In addition, `DBMS_CLOUD` you can load Data Pump dump files in the Cloud to your tables in Autonomous Database.

The package `DBMS_CLOUD` supports loading files from the following cloud services:

- Oracle Cloud Infrastructure Object Storage
- Azure Blob Storage
- Amazon S3
- Amazon S3-Compatible, including: Google Cloud Storage and Wasabi Hot Cloud Storage.
- GitHub Repository

Topics

- Create Credentials and Copy Data into an Existing Table
- Create Credentials and Load Data Pump Dump Files into an Existing Table
- Load JSON on Autonomous Database
- Monitor and Troubleshoot Loads
- List Credentials
- Delete Credentials
- Example Loading Data from a Fixed Width File

Create Credentials and Copy Data into an Existing Table

For data loading from files in the Cloud, you need to first store your object storage credentials in your Autonomous Database and then use the procedure `DBMS_CLOUD.COPY_DATA` to load data.

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

```
S,Direct Sales,Direct
T,Tele Sales,Direct
C,Catalog,Indirect
I,Internet,Indirect
P,Partners,Others
```

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

```sql
SET DEFINE OFF
BEGIN
DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adb_user@example.com',
    password => 'password'
);
```
This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for all data loads.

For detailed information about the parameters, see `CREATE_CREDENTIAL Procedure`.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See `Use Resource Principal to Access Oracle Cloud Infrastructure Resources` for more information.

**Note:**

Some tools like SQL*Plus and SQL Developer use the ampersand character (`&`) as a special character. If you have the ampersand character in your password use the `SET DEFINE OFF` command in those tools as shown in the example to disable the special character and get the credential created properly.

2. Load data into an existing table using the procedure `DBMS_CLOUD.COPY_DATA`. For example:

```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. The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
- `file_uri_list`: is a comma delimited list of the source files you want to load.
Create Credentials and Load Data Pump Dump Files into an Existing Table

For data loading you can also use Oracle Data Pump dump files as source files.

The source files for this load type must be exported from the source system using the ORACLE_DATAPUMP access driver in External Tables. See Unloading and Loading Data with the ORACLE_DATAPUMP Access Driver for details on exporting using the ORACLE_DATAPUMP access driver.

To load the data you first move the dump files that were exported using the ORACLE_DATAPUMP access driver to your Object Store and then use DBMS_CLOUD.COPY_DATA to load the dump files to an existing table in your database.

The source files in this example are the Oracle Data Pump dump files, exp01.dmp and exp02.dmp.

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

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

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for all data loads.

   For detailed information about the parameters, see CREATE_CREDENTIAL Procedure.
Note:

Some tools like SQL*Plus and SQL Developer use the ampersand character (&) as a special character. If you have the ampersand character in your password use the SET DEFINE OFF command in those tools as shown in the example to disable the special character and get the credential created properly.

2. Load data into an existing table using the procedure `DBMS_CLOUD.COPY_DATA`. For example:

```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' value 'datapump')
  );
END;
/
```

The parameters are:

- **table_name**: is the target table's name.
- **credential_name**: is the name of the credential created in the previous step. The credential name parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
- **file_uri_list**: is a comma delimited list of the Data Pump dump files you want to load.
- **format**: defines the options you can specify to describe the format of the source file. When you specify the type as 'datapump', the only other valid format parameter is 'rejectlimit'.

In this example, `namespace-string` is the Oracle Cloud Infrastructure object storage namespace and `bucketname` is the bucket name. See Understanding Object Storage Namespaces for more information.

For detailed information about the parameters, see COPY_DATA Procedure and COPY_DATA Procedure for Avro, ORC, or Parquet Files.
Load JSON on Autonomous Database

The PL/SQL procedure `DBMS_CLOUD.COPY_COLLECTION` provides support for loading JSON documents into SODA collections. The procedure `DBMS_CLOUD.COPY_DATA` provides support for loading JSON data into an existing table in Autonomous Database.

Topics

• About Loading JSON Documents
• Load a JSON File of Line-Delimited Documents into a Collection
• Load an Array of JSON Documents into a Collection
• Monitor and Troubleshoot COPY_COLLECTION Loads
• Objects That Extend JSON Scalars
• Create Credentials and Copy JSON Data into an Existing Table

About Loading JSON Documents

You load SODA collections into Autonomous Database using the PL/SQL procedure `DBMS_CLOUD.COPY_COLLECTION` and you load JSON data into a table using `DBMS_CLOUD.COPY_DATA`.

• `DBMS_CLOUD.COPY_COLLECTION` supports the following typical document loading procedures:
  – Loading line-delimited JSON into a collection. See Load a JSON File of Line-Delimited Documents into a Collection for this procedure.
  – Loading an array of JSON documents into a collection. See Load an Array of JSON Documents into a Collection for this procedure.
• `DBMS_CLOUD.COPY_DATA` supports the following for loading from JSON data in Object Store:
  – Create Credentials and Copy JSON Data into an Existing Table

Load a JSON File of Line-Delimited Documents into a Collection

For loading data from collections in the Cloud, you must first store your object storage credentials in your Autonomous Database and then use the procedure `DBMS_CLOUD.COPY_COLLECTION` to load documents into a collection.

This example loads JSON values from a line-delimited file and uses the JSON file `myCollection.json`. Each value, each line, is loaded into a collection on your database as a single document.

Here's an example of such a file. It has three lines, with one object per line. Each of those objects gets loaded as a separate JSON document.

```json
{ "name": "apple", "count": 20 }
{ "name": "orange", "count": 42 }
{ "name": "pear", "count": 10 }
```
Before loading the data from `myCollection.json` into your database, copy the file to your object store:

- Create a bucket in the object store. For example, create an Oracle Cloud Infrastructure Object Storage bucket from the Oracle Cloud Infrastructure Object Storage link, and then in your selected compartment click Create Bucket, or use a command such as the following OCI CLI command to create a bucket:

  ```
  oci os bucket create --name fruit_bucket -c <compartment id>
  ```

- Copy the JSON file to your object store bucket. For example use the following OCI CLI command to copy the JSON file to the `fruit_bucket` on Oracle Cloud Infrastructure Object Storage:

  ```
  oci os object put --bucket-name fruit_bucket \
  --file "myCollection.json"
  ```

Load the JSON file from object store into a collection named `fruit` on your database as follows:

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

   ```
   SET DEFINE OFF
   BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
     credential_name => 'DEF_CRED_NAME',
     username => 'adb_user@example.com',
     password => 'password'
   );
   END;
   /
   ```

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials, you can use the same credential name for loading all documents.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

   See `CREATE_CREDENTIAL Procedure` for detailed information about the parameters.

   **Note:**

   Some tools like SQL*Plus and SQL Developer use the ampersand character (`&`) as a special character. If you have the ampersand character in your password, then use the `SET DEFINE OFF` command in those tools as shown in the example to disable the special character, and get the credential created properly.
2. Load the data into a collection using the procedure `DBMS_CLOUD.COPY_COLLECTION`.

```plsql
BEGIN
    DBMS_CLOUD.COPY_COLLECTION(
        collection_name => 'fruit',
        credential_name => 'DEF_CRED_NAME',
        file_uri_list   =>
            'https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/
fruit_bucket/o/myCollection.json',
        format          =>
            JSON_OBJECT('recorddelimiter' value '''
              '\n''')  );
END;
/
```

The parameters are:

- **collection_name**: is the name of the target collection.
- **credential_name**: is the name of the credential created in the previous step. The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
- **file_uri_list**: is a comma delimited list of the source files that you want to load.
- **format**: defines the options that you can specify to describe the format of the source file. The format options `characterset`, `compression`, `ignoreblanklines`, `jsonpas`, `maxdocsize`, `recorddelimiter`, `rejectlimit`, `type`, `unpackarrays` are supported while loading JSON data. Any other formats specified will result in an error.

See `DBMS_CLOUD Package Format Options` for more information.

Where `namespace-string` is the Oracle Cloud Infrastructure object storage namespace and `fruit_bucket` is the bucket name. See `Understanding Object Storage Namespaces` and `Overview of Object Storage` for more information.

For detailed information about the parameters, see `COPY_COLLECTION Procedure`.

The collection `fruit` on your database now contains one document for each line in the file `myCollection.json`.

**Load an Array of JSON Documents into a Collection**

To load data from collections in the Cloud, you first store your object storage credentials in your Autonomous Database and then use PL/SQL procedure `DBMS_CLOUD.COPY_COLLECTION` to load documents into a collection. This topic explains how to load documents to your database from a JSON array in a file.

**Note:**
You can also load documents from a JSON array in a file into a collection using SODA for REST. See `Load Purchase-Order Sample Data Using SODA for REST`. 
This example uses the JSON file `fruit_array.json`. The following shows the contents of the file `fruit_array.json`:

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

Before loading data into Autonomous Database, copy the data to your object store as follows:

- Create a bucket in the object store. For example, create an Oracle Cloud Infrastructure Object Store bucket from the Oracle Cloud Infrastructure Object Storage link, in your selected Compartment, by clicking **Create Bucket**, or use a command line tool such as the following OCI CLI command:

  ```
  oci os bucket create -name json_bucket -c <compartment id>
  ```

- Copy the JSON file to the object store. For example, the following OCI CLI command copies the JSON file `fruit_array.json` to the object store:

  ```
  oci os object put --bucket-name json_bucket --file "fruit_array.json"
  ```

Load the JSON file from object store into a SODA collection named `fruit2` on your database:

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

   ```
   SET DEFINE OFF
   BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
     credential_name => 'DEF_CRED_NAME',
     username => 'adb_user@example.com',
     password => 'password'
   );
   END;
   /
   ```

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials, you can use the same credential name for loading all documents.

   See **CREATE_CREDENTIAL Procedure** for detailed information about the parameters.
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",
    "maxdocsize" : "10240000"
  );
END;
/
```

In this example you load a single JSON value which occupies the whole file, so there is no need to specify a record delimiter. To indicate that there is no record delimiter, you can use a character that does not occur in the input file. For this example, to indicate that there is no delimiter, the control character 0x01 (SOH) is set to load the JSON documents into a collection. Thus, you specify a value for the `recorddelimiter` that does not occur in the JSON file. For example, you can use value "0x'01'" because this character does not occur directly in JSON text.

When `unpackarrays` parameter for format value is set to `TRUE`, the array of documents is loaded as individual documents rather than as an entire array. The unpacking of array elements is however limited to single level. If there are nested arrays in the documents, those arrays are not unpacked.

The parameters are:

- `collection_name` is the name of the target collection.
- `credential_name` is the name of the credential created in the previous step. The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
- `file_uri_list` is a comma delimited list of the source files that you want to load.
- `format` defines the options that you can specify to describe the format of the source file. The format options `characterset`, `compression`, `ignoreblanklines`, `jsonpah`, `maxdocsize`, `recorddelimiter`, `rejectlimit`, `type`, `unpackarrays` are supported for 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.COPY_COLLECTION using the format option unpackarrays recognizes array values in the source and instead of loading the data as a single document, as it would by default, the data is loaded in the collection fruit2 with each value in the array as a single document.

Monitor and Troubleshoot COPY_COLLECTION Loads

All data load operations you perform using the PL/SQL package DBMS_CLOUD are logged in the tables dba_load_operations and user_load_operations. Use these tables to monitor loading with DBMS_CLOUD.COPY_COLLECTION.

- dba_load_operations shows all load operations
- user_load_operations shows the load operations in your schema

You can query these tables to see information about ongoing and completed data loads. For example, the following SELECT statement with a WHERE clause predicate on the TYPE column shows load operations of the type COPY:

```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>
<th>LOGFILE_TABLE</th>
<th>BADFILE_TABLE</th>
</tr>
</thead>
</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:

```sql
SELECT * FROM COPY2_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:

```sql
SELECT * FROM COPY2_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.
Objects That Extend JSON Scalars

Native binary JSON data (OSON format) extends the JSON language by adding scalar types, such as date, that correspond to SQL types and are not part of the JSON standard. Oracle Database also supports the use of textual JSON objects that represent JSON scalar values, including such nonstandard values.

When you create native binary JSON data from textual JSON data that contains such extended objects, they can optionally be replaced with corresponding (native binary) JSON scalar values.

An example of an extended object is \{"$numberDecimal":31\}. It represents a JSON scalar value of the nonstandard type decimal number, and when interpreted as such it is replaced by a decimal number in native binary format.

For example, when you use the JSON data type constructor, JSON, if you use keyword EXTENDED then recognized extended objects in the textual input are replaced with corresponding scalar values in the native binary JSON result. If you do not include keyword EXTENDED then no such replacement occurs; the textual extended JSON objects are simply converted as-is to JSON objects in the native binary format.

In the opposite direction, when you use Oracle SQL function json_serialize to serialize binary JSON data as textual JSON data (VARCHAR2, CLOB, or BLOB), you can use keyword EXTENDED to replace (native binary) JSON scalar values with corresponding textual extended JSON objects.

Note:

If the database you use is an Oracle Autonomous Database then you can use PL/SQL procedure DBMS_CLOUD.copy_collection to create a JSON document collection from a file of JSON data such as that produced by common NoSQL databases, including Oracle NoSQL Database.

If you use ejson as the value of the type parameter of the procedure, then recognized extended JSON objects in the input file are replaced with corresponding scalar values in the resulting native binary JSON collection. In the other direction, you can use function json_serialize with keyword EXTENDED to replace scalar values with extended JSON objects in the resulting textual JSON data.

These are the two main use cases for extended objects:

- **Exchange (import/export):**
  - Ingest existing JSON data (from somewhere) that contains extended objects.
  - Serialize native binary JSON data as textual JSON data with extended objects, for some use outside the database.

- Inspection of native binary JSON data: see what you have by looking at corresponding extended objects.
For exchange purposes, you can ingest JSON data from a file produced by common NoSQL databases, including Oracle NoSQL Database, converting extended objects to native binary JSON scalars. In the other direction, you can export native binary JSON data as textual data, replacing Oracle-specific scalar JSON values with corresponding textual extended JSON objects.

As an example of inspection, consider an object such as {"dob": "2000-01-02T00:00:00"} as the result of serializing native JSON data. Is "2000-01-02T00:00:00" the result of serializing a native binary value of type date, or is the native binary value just a string? Using jsonSerialize with keyword EXTENDED lets you know.

The mapping of extended object fields to scalar JSON types is, in general, many-to-one: more than one kind of extended JSON object can be mapped to a given scalar value. For example, the extended JSON objects {"$numberDecimal":"31"} and {"$numberLong":"31"} are both translated as the value 31 of JSON-language scalar type number, and item method type() returns number for each of those JSON scalars.

Item method type() reports the JSON-language scalar type of its targeted value. Some scalar values are distinguishable internally, even when they have the same scalar type. This generally allows function jsonSerialize (with keyword EXTENDED) to reconstruct the original extended JSON object. They are distinguished internally either by using different SQL types to implement them or by tagging them with the kind of extended JSON object from which they were derived.

When jsonSerialize reconstructs the original extended JSON object the result is not always textually identical to the original, but it is always semantically equivalent. For example, {"$numberDecimal":"31"} and {"$numberDecimal":31} are semantically equivalent, even though the field values differ in type (string and number). They are translated to the same internal value, and each is tagged as being derived from a $numberDecimal extended object (same tag). But when serialized, the result for both is {"$numberDecimal":31}. Oracle always uses the most directly relevant type for the field value, which in this case is the JSON-language value 31, of scalar type number.

Note:

There are two cases where the type of the original extended object can be lost when deriving the internal binary-JSON value.

- An extended object with field $numberInt is translated to an Oracle SQL NUMBER internal value, with no tag. Serializing that value produces a standard JSON-language value of type number. There is no loss in the numerical value; the only loss is the information that the original textual data was a $numberInt extended object.

- Use of field $numberDecimal with infinite, very small, very large, or not-a-number values is unsupported, and results in undefined behavior. Do not use a string value that represents positive infinity ("Infinity" or "Inf"), negative infinity ("-Infinity" or "-Inf"), or an unknown value (not a number, "Nan") with $numberDecimal — instead, use $numberDouble with such values.
You can generally go back and forth between native binary JSON data and textual JSON data without loss of information. However, comparison (and hence indexing) of data in SQL requires that you stay within the same type family.

You can use item method `type()` to identify the type family of a JSON value (but not the exact type within a family), which makes it useful for purposes of comparison or indexing.

You can compare JSON values only within each of the following type families.

- **Floating-point number types**: double and float (from extended objects with $numberDouble or $numberFloat).
  - Item method `type()` reports values in this family as `double` or `float`.

- **Decimal number types** (from extended objects with $numberInt, $numberDecimal, or $numberLong).
  - Item method `type()` reports values in this family as `number`.

- **Binary types**, including identifiers (from extended objects with $binary, $oid, $rawhex or $rawid).
  - Item method `type()` reports values in this family as `binary`.

- **Date and time point types** (from extended objects with $date, $oracleDate, $oracleTimestamp or $oracleTimestampTZ).
  - Item method `type()` reports values in this family as `date` or `timestamp`. It reports a timestamp-with-timezone value (from extended objects with $oracleTimestampTZ) as `timestamp`.
  - A $date field has a timestamp-with-timezone value, because it allows fractional seconds, and the value is given for Coordinated Universal Time (UTC).

- **Date and time interval types** (from extended objects with $intervalDaySecond or $intervalYearMonth).
  - Item method `type()` reports values in this family as `daysecondInterval` or `yearmonthInterval`.

- **JSON string type**
  - Item method `type()` reports values in this family as `string`.

- **JSON null type**
  - Item method `type()` reports values in this family as `null`.

- **JSON Boolean type**
  - Item method `type()` reports values in this family as `boolean`.

Table 3-1 presents correspondences among the various types used. It maps across types of extended objects used as input, types reported by item method `type()`, SQL types used internally, standard JSON-language types used as output by function `json_serialize`, and types of extended objects output by `json_serialize` when keyword `EXTENDED` is specified.
Table 3-1  Extended JSON Object Type Relations

<table>
<thead>
<tr>
<th>Extended Object Type (Input)</th>
<th>Oracle JSON Scalar Type (Reported by type())</th>
<th>SQL Scalar Type</th>
<th>Standard JSON Scalar Type (Output)</th>
<th>Extended Object Type (Output)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$numberDouble with value a JSON number, a string representing the number, or one of these strings: &quot;Infinity&quot;, &quot;-Infinity&quot;, &quot;Inf&quot;, &quot;-Inf&quot;, &quot;Nan&quot;</td>
<td>double</td>
<td>BINARY_DOUBLE</td>
<td>number</td>
<td>$numberDouble with value a JSON number or one of these strings: &quot;Inf&quot;, &quot;-Inf&quot;, &quot;Nan&quot;</td>
</tr>
<tr>
<td>$numberFloat with value the same as for $numberDouble</td>
<td>float</td>
<td>BINARY_FLOAT</td>
<td>number</td>
<td>$numberFloat with value the same as for $numberDouble</td>
</tr>
<tr>
<td>$numberDecimal with value the same as for $numberDouble</td>
<td>number</td>
<td>NUMBER</td>
<td>number</td>
<td>$numberDecimal with value the same as for $numberDouble</td>
</tr>
<tr>
<td>$numberInt with value a signed 32-bit integer or a string representing the number</td>
<td>number</td>
<td>NUMBER</td>
<td>number</td>
<td>$numberInt with value the same as for $numberDouble</td>
</tr>
<tr>
<td>$numberLong with value a JSON number or a string representing the number</td>
<td>number</td>
<td>NUMBER</td>
<td>number</td>
<td>$numberLong with value the same as for $numberDouble</td>
</tr>
<tr>
<td>$binary with value one of these:</td>
<td>binary</td>
<td>BLOB or RAW</td>
<td>string</td>
<td>One of the following:</td>
</tr>
<tr>
<td>• a string of base-64 characters</td>
<td></td>
<td></td>
<td>Conversion is equivalent to the use of SQL function rawtohex.</td>
<td></td>
</tr>
<tr>
<td>• An object with fields base64 and subType, whose values are a string of base-64 characters and the number 0 (arbitrary binary) or 4 (UUID), respectively</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>When the value is a string of base-64 characters, the extended object can also have field $subtype with value 0 or 4, expressed as a one-byte integer (0-255) or a 2-character hexadecimal string, representing such an integer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$oid with value a string of 24 hexadecimal characters</td>
<td>binary</td>
<td>RAW(12)</td>
<td>string</td>
<td>$oid with value a string of 24 hexadecimal characters</td>
</tr>
<tr>
<td>$rawhex with value a string with an even number of hexadecimal characters</td>
<td>binary</td>
<td>RAW</td>
<td>string</td>
<td>$rawhex with value a string with an even number of hexadecimal characters</td>
</tr>
<tr>
<td>$binary with a string of base-64 characters</td>
<td></td>
<td></td>
<td>$binary with a string of base-64 characters, right-padded with = characters</td>
<td></td>
</tr>
</tbody>
</table>
Table 3-1 (Cont.) Extended JSON Object Type Relations

<table>
<thead>
<tr>
<th>Extended Object Type (Input)</th>
<th>Oracle JSON Scalar Type (Reported by type())</th>
<th>SQL Scalar Type</th>
<th>Standard JSON Scalar Type (Output)</th>
<th>Extended Object Type (Output)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$rawid with value a string of 24 or 32 hexadecimal characters</td>
<td>binary</td>
<td>RAW</td>
<td>string</td>
<td>$rawid</td>
</tr>
<tr>
<td>$oracleDate with value an ISO 8601 date string</td>
<td>date</td>
<td>DATE</td>
<td>string</td>
<td>$oracleDate with value an ISO 8601 date string</td>
</tr>
<tr>
<td>$oracleTimestamp with value an ISO 8601 timestamp string</td>
<td>timestamp</td>
<td>TIMESTAMP</td>
<td>string</td>
<td>$oracleTimestamp with value an ISO 8601 timestamp string</td>
</tr>
<tr>
<td>$oracleTimestampTZ with value an ISO 8601 timestamp string with a numeric time zone offset or with Z</td>
<td>timestamp</td>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>string</td>
<td>$oracleTimestampTZ with value an ISO 8601 timestamp string with a numeric time zone offset or with Z</td>
</tr>
<tr>
<td>$date with value one of the following:</td>
<td>timestamp</td>
<td>TIMESTAMP WITH TIME ZONE</td>
<td>string</td>
<td>$oracleTimestampTZ with value an ISO 8601 timestamp string with a numeric time zone offset or with Z</td>
</tr>
<tr>
<td>• An integer millisecond count since January 1, 1990</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• An ISO 8601 timestamp string</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• An object with field numberLong with value an integer millisecond count since January 1, 1990</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$intervalDaySecond with value an ISO 8601 interval string as specified for SQL function to_dsinterval</td>
<td>dayseconds</td>
<td>INTERVAL DAY TO SECOND</td>
<td>string</td>
<td>$intervalDaySecond with value an ISO 8601 interval string as specified for SQL function to_dsinterval</td>
</tr>
<tr>
<td>$intervalYearMonth with value an ISO 8601 interval string as specified for SQL function to_yminterval</td>
<td>yearmonths</td>
<td>INTERVAL YEAR TO MONTH</td>
<td>string</td>
<td>$intervalYearMonth with value an ISO 8601 interval string as specified for SQL function to_yminterval</td>
</tr>
</tbody>
</table>

1 The string values are interpreted case-insensitively. For example, "NAN" "nan", and "nAn" are accepted and equivalent, and similarly "INF", "Infinity", and "INF". Infinitely large ("Infinity" or "Inf") and small ("-Infinity" or "-Inf") numbers are accepted with either the full word or the abbreviation.

2 On output, only these string values are used — no full-word infinity or letter-case variants.

See Also:
IEEE Standard for Floating-Point Arithmetic (IEEE 754)
Create Credentials and Copy JSON Data into an Existing Table

Use `DBMS_CLOUD.COPY_DATA` to load JSON data in the cloud into a table.

The source file in this example is a JSON data file.

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

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

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for all data loads.

   For detailed information about the parameters, see `CREATE_CREDENTIAL` Procedure.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

2. Load JSON data into an existing table using the procedure `DBMS_CLOUD.COPY_DATA`.

   For example:

   ```sql
   CREATE TABLE WEATHER2
     (WEATHER_STATION_ID VARCHAR2(20),
      WEATHER_STATION_NAME VARCHAR2(50));
   /
   BEGIN
     DBMS_CLOUD.COPY_DATA(
       table_name      => 'WEATHER2',
       credential_name => 'DEF_CRED_NAME',
       file_uri_list   => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/jsonfiles*',
       format          => JSON_OBJECT('type' value 'json',
                                      'columnpath' value '[$.WEATHER_STATION_ID, $.WEATHER_STATION_NAME]')
     );
   END;
   /
   ```
The parameters are:

- **table_name**: is the target table's name.
- **credential_name**: is the name of the credential created in the previous step.
- **file_uri_list**: is a comma delimited list of the source files you want to load. You can use wildcards in the file names in your URIs. The character "*" can be used as the wildcard for multiple characters, the character "?" can be used as the wildcard for a single character.
- **format**: for `DBMS_CLOUD.COPY_DATA` with JSON data, the type is `json`. Specify other format values to define the options to describe the format of the JSON source file. See `DBMS_CLOUD` Package Format Options for more information.

In this example, `namespace-string` is the Oracle Cloud Infrastructure object storage namespace and `bucketname` is the bucket name. See Understanding Object Storage Namespaces for more information.

For detailed information about the parameters, see `COPY_DATA Procedure`.

### Monitor and Troubleshoot Loads

All data load operations done using the PL/SQL package `DBMS_CLOUD` are logged in the tables `dba_load_operations` and `user_load_operations`:

- **dba_load_operations**: shows all load operations.
- **user_load_operations**: shows the load operations in your schema.

Query these tables to see information about ongoing and completed data loads. For example:

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

```
TABLE_NAME | OWNER_NAME | TYPE   | STATUS    | START_TIME              | UPDATE_TIME            | LOGFILE_TABLE | BADFILE_TABLE
------------|------------|--------|-----------|------------------------|------------------------|---------------|-------------------
CHANNELS    | SH         | COPY   | COMPLETED | 06-NOV-20 01.55.19.3   | 06-NOV-20 01.55.28.2   | COPY$21_LOG   | COPY$21_BAD
```

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:

```sql
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:

```sql
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 ORC, Parquet, or Avro File Loading**

As with other data files, ORC, Parquet, and Avro data loads generate logs that are viewable in the tables `dba_load_operations` and `user_load_operations`. Each load operation adds a record to `dba[<user>]_load_operations` that indicates the table containing the logs.

The log table provides summary information about the load.

Note:

For ORC, Parquet, or Avro files, when the `format` parameter `type` is set to the value `orc`, `parquet` or `avro` the `BADFILE_TABLE` table is always empty.

- **PRIMARY KEY** constraint errors throw an ORA error.
- If data for a column encounters a conversion error, for example, the target column is not large enough to hold the converted value, the value for the column is set to `NULL`. This does not produce a rejected record.

**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:

```sql
SELECT credential_name, username, comments FROM all_credentials;
```

<table>
<thead>
<tr>
<th>CREDENTIAL_NAME</th>
<th>USERNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMENTS</td>
<td>---------------------</td>
</tr>
<tr>
<td>ADB_TOKEN</td>
<td><a href="mailto:user_name@example.com">user_name@example.com</a></td>
</tr>
<tr>
<td>{&quot;comments&quot;:&quot;Created via DBMS_CLOUD.create_credential&quot;}</td>
<td></td>
</tr>
</tbody>
</table>
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 remove credentials with `DBMS_CLOUD.DROP_CREDENTIAL`.

For example, to remove the credential named `DEF_CRED_NAME`, run the following command:

```sql
BEGIN
    DBMS_CLOUD.DROP_CREDENTIAL('DEF_CRED_NAME');
END;
```

For more information about the `DBMS_CLOUD` procedures and parameters, see `DBMS_CLOUD Subprograms and REST APIs`.

Example Loading Data from a Fixed Width File

Load a Fixed-Width File into a New Table

This provides an example using `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` to load data from a fixed-width source file to an external table.

For this example, the fixed-width source file has the following data:

<table>
<thead>
<tr>
<th>INDEX</th>
<th>TABLE</th>
<th>VALID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>INDEX</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>1</td>
<td>INDEX</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>2</td>
<td>INDEX</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>3</td>
<td>INDEX</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>4</td>
<td>TABLE</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>5</td>
<td>TABLE</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>6</td>
<td>CLUSTER</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>7</td>
<td>INDEX</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>8</td>
<td>INDEX</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>9</td>
<td>INDEX</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>10</td>
<td>INDEX</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>11</td>
<td>TABLE</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>12</td>
<td>INDEX</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>13</td>
<td>INDEX</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>14</td>
<td>TABLE</td>
<td>2001272020012720200127VALID</td>
</tr>
<tr>
<td>15</td>
<td>INDEX</td>
<td>2001272020012720200127VALID</td>
</tr>
</tbody>
</table>

1. From the console, select the compartment for your Autonomous Database, and then select the link to your Autonomous Database to open the console.
2. On the Autonomous Database Details page click **Database Actions**.

3. Within the **Development** section of the **Database Actions Launchpad**, select the **SQL** tile.

4. Within the SQL Worksheet, enter and execute the following code:

   ```sql
   BEGIN
   DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
       table_name      => '<YOUR_TABLE_NAME>'
   ,
     credential_name => '<YOUR_CREDENTIAL_NAME>'
   ,
     file_uri_list   => '<YOUR_ORACLE_OBJECT_STORE_URL>'
   ,
     format          => json_object('trimspaces' value 'rtrim','skipheaders' value '1', 'dateformat' value 'YYYYMMDD')
   ,
     field_list      => 'object_id      (1:3)   char
                    , object_name    (4:14)  char
                    , object_type    (15:39)  char
                    , created_date1  (40:45)  date mask "YYMMDD"
                    , created_date2  (46:53)  date
                    , last_ddl_time  (54:61)  date
                    , status         (62:71)'
   ,
     column_list     => 'object_id      number
                    , object_name    varchar2(30)
                    , object_type    varchar2(25)
                    , status         varchar2(10)
                    , created_date1  date
                    , created_date2  date
                    , last_ddl_time  date');
   END;
   /
   ``
When a load or import operation results in the following timezone related error, you should restart your Autonomous Database and try again:

ORA-39405: Oracle Data Pump does not support importing from a source database with TSTZ version \( n+1 \)
into a target database with TSTZ version \( n \).

See Manage Time Zone File Version on Autonomous Database for more information on this timezone related error.

Topics

- Export Your Existing Oracle Database to Import into Autonomous Database
- 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 Database

Use Oracle Data Pump Export to export your existing Oracle Database to migrate to Autonomous Database using Oracle Data Pump Import.

Oracle recommends using Oracle Data Pump schema mode to migrate your database to Autonomous Database. You can list the schemas you want to export by using the schema parameter.

For a faster migration, export your schemas into multiple Data Pump files and use parallelism. You can specify the dump file name format you want to use with the dumpfile parameter. Set the parallel parameter to at least the number of CPUs you have in your database.

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

```bash
exclude=cluster,indextype,db_link
parallel=n
schemas=schema_name
dumpfile=export%u.dmp
```

The exclude parameters ensure that these object types are not exported.

With encryption_pwd_prompt=yes Oracle Data Pump export prompts for an encryption password to encrypt the dump files.

The following example exports the SH schema from a source Oracle Database for migration to a database with 16 CPUs:

```bash
expdp sh/sh@orcl \
exclude=cluster,indextype,db_link \
parallel=16 \
schemas=sh \
dumpfile=export%u.dmp \
encryption_pwd_prompt=yes
```
Note:
If during the export with `expdp` you use the `encryption_pwd_prompt=yes` parameter then also use `encryption_pwd_prompt=yes` with your import and input the same password at the `impdp` prompt to decrypt the dump files (remember the password you supply during export). The maximum length of the encryption password is 128 bytes.

You can use other Data Pump Export parameters, like compression, depending on your requirements. For more information on Oracle Data Pump Export see Oracle Database Utilities.

Import Data Using Oracle Data Pump Version 18.3 or Later

Oracle recommends using the latest Oracle Data Pump version for importing data from Data Pump files into your Autonomous Database, as it contains enhancements and fixes for a better experience.

Download the latest version of Oracle Instant Client, which includes Oracle Data Pump, for your platform from Oracle Instant Client Downloads. See the installation instructions on the platform install download page for the installation steps required after you download Oracle Instant Client.

In Oracle Data Pump version 18.3 and later, the `credential` argument authenticates Data Pump to the Cloud Object Storage service you are using for your source files. The `dumpfile` argument is a comma delimited list of URLs for your Data Pump files.

In Oracle Data Pump, if your source files reside on Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure native URIs, or Swift URIs. See DBMS_CLOUD Package File URI Formats for details on these file URI formats.

Importing with Oracle Data Pump and Setting `credential` Parameter

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

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

   ```bash
   impdp admin/password@db2022adb_high \
   directory=data_pump_dir \
   ```
credential=def_cred_name \
  dumpfile= https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/bucketname/o/export%u.dmp \
  parallel=16 \ 
  encryption_pwd_prompt=yes \
  exclude=cluster,indextype,db_link

**Note:**

If during the export with expdp you used the encryption_pwd_prompt=yes parameter then use encryption_pwd_prompt=yes and input the same password at the impdp prompt that you specified during the export.

In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

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 Database as shown in the example.

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

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

In this example the following are excluded during the Data Pump Import:

- Clusters
- Indextypes
- Database links

**Note:**

To perform a full import or to import objects that are owned by other users, you need the DATAPUMP_CLOUD_IMP role.

You can also use Data Pump Import to import SODA collections on Autonomous Database. See Import SODA Collection Data Using Oracle Data Pump Version 19.6 or Later for more information.

For information on disallowed objects in Autonomous Database, see SQL Commands.

For detailed information on Oracle Data Pump Import parameters see Oracle Database Utilities.
Import Data Using Oracle Data Pump (Versions 12.2.0.1 and Earlier)

You can import data from Data Pump files into your Autonomous Database using Data Pump client versions 12.2.0.1 and earlier by setting the `default_credential` parameter.

Data Pump Import versions 12.2.0.1 and earlier do not have the `credential` parameter. If you are using an older version of Data Pump Import you need to define a default credential property for Autonomous Database and use the `default_credential` keyword in the `dumpfile` parameter.

In Oracle Data Pump, if your source files reside on Oracle Cloud Infrastructure Object Storage you can use the Oracle Cloud Infrastructure native URIs, or Swift URIs. See DBMS_CLOUD Package File URI Formats for details on these file URI formats.

Importing with Older Oracle Data Pump Versions and Setting `default_credential`

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

   ```sql
   BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
     credential_name => 'DEF_CRED_NAME',
     username => 'adb_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 Database, 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@db2022adb_high \
   directory=data_pump_dir \
   dumpfile=default_credential:https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/bucketname/o/export%u.dmp \
   parallel=16 \
   encryption_pwd_prompt=yes \
   exclude=cluster,indextype,db_link
   ```
Note:

If during the export with expdp you used the encryption_pwd_prompt=yes parameter then use encryption_pwd_prompt=yes and input the same password at the impdp prompt that you specified during the export.

In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

For the best import performance use the HIGH database service for your import connection and set the PARALLEL parameter to the number of OCPUs in your Autonomous Database as shown in the example.

For information on which database service name to connect to run Data Pump Import, Manage Concurrency and Priorities on Autonomous Database.

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

In this example the following are excluded during the Data Pump Import:

- Clusters
- Indextypes
- Database links

Note:

To perform a full import or to import objects that are owned by other users, you need the DATAPUMP_CLOUD_IMP role.

You can also use Data Pump Import to import SODA collections on Autonomous Database. See Import SODA Collection Data Using Oracle Data Pump Version 19.6 or Later for more information.

For information on disallowed objects in Autonomous Database, see SQL Commands.

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:

```
BEGIN
    DBMS_CLOUD.PUT_OBJECT(
        credential_name => 'DEF_CRED_NAME',
```

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

For more information, see DBMS_CLOUD Subprograms and REST APIs.

Load Data from Local Files with Oracle Database Actions

Using Oracle Database Actions, from the Worksheet page, you can load data from local files into an existing table.

Topics

• Load Data into Existing Autonomous Database Table with Oracle Database Actions

Load Data into Existing Autonomous Database Table with Oracle Database Actions

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

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

To upload data from local files to an existing table with Database Actions, do the following:

1. Access Database Actions from the Oracle Cloud Infrastructure Console or with the Database Actions link provided to you.

2. To import data, in Database Actions, under Development click SQL.

This shows an SQL worksheet.
3. In the Navigator, right-click the table where you want to load data.

4. In the menu select **Data loading → Upload Data**...
   For example, select the **SALES** table, right-click, and select **Data loading → Upload Data**...

   ![Oracle Database Actions | SQL](image)

   This shows the Import data dialog:
5. In the Import data dialog you can either drag and drop files or click **Select files** to show a browser to select the files to import.

6. Complete the mapping for the columns you are importing. There are a number of options for column mapping. Click (Show/Hide options) icon to show the data import and format options to change column names, skip rows, rows to load, and various other options.

   Click **Apply** to apply the options you select.

7. When you finish selecting format and mapping options, click **Next** to preview the column mapping.

   If there is a problem at this stage, information shows with more details, such as: 2 pending actions. This means you need to correct or fix the source file data before you import.

8. Click **Next**.
9. Click **Next** to review the column mapping.

This shows the **Review** page to review the source columns and target columns for the import:

10. Click **Finish**.

11. Click **OK** to confirm the import.

Depending on the size of the data file you are importing, the import may take some time.

Database Actions provides history to show the status of the import and to allow you to review the results or errors associated with the import operation.

For a detailed summary of the upload process, right-click the table in the **Navigator** tab, select **Data loading**, and then select **Loaded Data**. A summary of the data loaded is displayed in the Loaded data summary dialog.

If any data failed to load, you can view the number of rows in the Failed Rows column. Click the column and a dialog is displayed showing the failed rows.

In the Loaded data summary dialog, you can also search for files loaded by schema name, table name, or file name. To remove the loaded files, click the Delete icon.

See Uploading Data from Local Files for more information on using Database Actions to upload data.

**Use Oracle GoldenGate to Replicate Data to Autonomous Database**

You can replicate data to Oracle Autonomous Database using Oracle GoldenGate.

Oracle GoldenGate Capture for Oracle Autonomous Database supports the following:

- Replication for different use cases: Report Offloading, Active-Active, Cloud to Cloud, and Cloud to on-premise.
- Inter-region and cross-region replication: Replicate data between different Oracle Cloud data centers around the world.
• Replicate between targets: Replicate from an Autonomous Database to any target database or platform that Oracle GoldenGate supports, including to other Oracle Autonomous Database environments.

See Using Oracle GoldenGate with Autonomous Database for more information.

Load Data from Local Files Using SQL*Loader

Instead of using SQL*Loader Oracle recommends loading data from the Cloud Object Storage for better performance and enhanced functionality.

For information on loading from Cloud Object Store, see Load Data from Files in the Cloud.

Depending on your workload type, note the following:

• **Data Warehouse:** If you use SQL*Loader to load data, note that Autonomous Database does not gather optimizer statistics for your load and you need to gather optimizer statistics manually as explained in Manage Optimizer Statistics on Autonomous Database. Autonomous Database gathers optimizer statistics automatically for tables loaded with direct path operations issued in SQL (direct path load operations that bypass the SQL data processing, such as SQL*Loader direct path, do not collect statistics).

• **Transaction Processing** or **JSON Database:** If you use SQL*Loader to load data, note that Autonomous Database does not gather optimizer statistics for your load and you need to gather optimizer statistics manually as explained in Manage Optimizer Statistics on Autonomous Database or wait for the automatic statistic gathering task to kick in.

For detailed information on SQL*Loader see, Oracle Database Utilities.
Querying External Data with Autonomous Database

Describes packages and tools to query and validate data with Autonomous Database.

External data is not managed by the database; however, you can use `DBMS_CLOUD` procedures to query your external data. Although queries on external data will not be as fast as queries on database tables, you can use this approach to quickly start running queries on your external source files and external data. Depending on the type of external table, you can validate external data using the `DBMS_CLOUD` validation procedures. The data validation procedures let you validate the source files for an external table so that you can identify problems and either correct the data in the external table or exclude invalid data before you use the data.

**Note:**

If you are not using `ADMIN` user, ensure the user has the necessary privileges for the operations the user needs to perform. See Manage User Privileges on Autonomous Database - Connecting with a Client Tool for more information.

Topics

- Query External Data
- Query External Data with ORC, Parquet, or Avro Source Files
- Query External Tables with Partitioning Specified in Source Files
- Query External Partitioned Data (with Partitioning Clause)
- Query Hybrid Partitioned Data
- Query External Data Pump Dump Files
- Query Big Data Service Hadoop (HDFS) Data from Autonomous Database
- Query External Data with Data Catalog
- 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 Database, and then create an external table using the PL/SQL procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`.

The procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` supports external files in the supported cloud object storage services, including:

- Oracle Cloud Infrastructure Object Storage
- Azure Blob Storage
- Amazon S3
- Amazon S3-Compatible, including: Oracle Cloud Infrastructure Object Storage, Google Cloud Storage, and Wasabi Hot Cloud Storage.
- GitHub Repository

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

```
S, Direct Sales, Direct
T, Tele Sales, Direct
C, Catalog, Indirect
I, Internet, Indirect
P, Partners, Others
```

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

   For example:

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

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for creating external tables.

   See `CREATE_CREDENTIAL Procedure` for information about the `username` and `password` parameters for different object storage services.

2. Create an external table on top of your source files using the procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`.

   The procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` supports external files in the supported cloud object storage services. The credential is a table level property; therefore, the external files must be on the same object store.
For example:

```
BEGIN
  DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
    table_name =>'CHANNELS_EXT',
    credential_name =>'DEF_CRED_NAME',
    file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/'
      namespace-string/b/bucketname/o/channels.txt',
    format => json_object('delimiter' value ','),
    column_list =>'"CHANNEL_ID 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. The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
- `file_uri_list`: is a comma delimited list of the source files you want to query.
- `format`: defines the options you can specify to describe the format of the source file.
- `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.

External Table Metadata Columns

The external table metadata helps you determine where data is coming from when you perform a query.

The external tables you create with DBMS_CLOUD.CREATE_EXTERNAL_TABLE, DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE, or DBMS_CLOUD.CREATE_HYBRID_PART_TABLE include two invisible columns file$path and file$name. These columns help identify which file a record is coming from.

- file$path: Specifies the file path text up to the beginning of the object name.
- file$name: Specifies the object name, including all the text that follows the final "/".

For example:

```
SELECT genre_id, name, file$name, file$path FROM ext_genre
WHERE rownum <= 2;
```

<table>
<thead>
<tr>
<th>genre_id</th>
<th>name</th>
<th>file$name</th>
<th>file$path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Action</td>
<td>genre.csv</td>
<td><a href="https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/moviestream_gold/o/genre">https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/moviestream_gold/o/genre</a></td>
</tr>
<tr>
<td>2</td>
<td>Adventure</td>
<td>genre.csv</td>
<td><a href="https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/moviestream_gold/o/genre">https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/moviestream_gold/o/genre</a></td>
</tr>
</tbody>
</table>

See Invisible Columns for more information on invisible columns.

Query External Data with ORC, Parquet, or Avro Source Files

Autonomous Database makes it easy to access ORC, Parquet, or Avro data stored in object store using external tables. ORC, Parquet, and Avro sources have metadata embedded in them and the DBMS_CLOUD.CREATE_EXTERNAL_TABLE procedure can utilize this metadata to simplify the creation of external tables.

You don’t need to know the structure of the data, DBMS_CLOUD can examine the file and convert ORC, Parquet, or Avro contents into the equivalent Oracle columns and data types. You only need to know the location of the data in object store, specify its type, ORC, Parquet, or Avro, and have credentials to access the source file on your object store.

Note:
The steps to use external tables are very similar for ORC, Parquet, and Avro. These steps show working with a Parquet format source file.
The source file in this example, `sales_extended.parquet`, contains Parquet format data. To query this file in Autonomous Database, do the following:

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

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

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for creating external tables.

   See `CREATE_CREDENTIAL Procedure` for information about the `username` and `password` parameters for different object storage services.

2. Create an external table for ORC, Parquet, or Avro on top of your source files using the procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`.

   The procedure `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` supports external files in the supported cloud object storage services, including: Oracle Cloud Infrastructure Object Storage, Azure Blob Storage, Amazon S3, and Amazon S3-Compatible, including: Oracle Cloud Infrastructure Object Storage, Google Cloud Storage, and Wasabi Hot Cloud Storage. The credential is a table level property; therefore, the external files must be on the same object store.

   By default, the columns created in the external table automatically map their data types to Oracle data types for the fields found in the source files and the external table column names match the source field names.

   ```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. The **credential_name** parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.

• **file_uri_list**: is a comma delimited list of the source files you want to query.

• **format**: defines the options to describe the format of the source file. For a Parquet file, use the **format** parameter to specify the type parquet. For an Avro file use the **format** parameter to specify the type avro. For an ORC file use the **format** parameter to specify the type orc.

In this example, **namespace-string** is the Oracle Cloud Infrastructure object storage namespace and **bucketname** is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

By default the **format schema** parameter is set and the columns and data types are derived automatically and the fields in the source match the external table columns by name. Source data types are converted to the external table column Oracle data types according to the **DBMS_CLOUD** mapping for ORC, Parquet, or Avro data types. The valid **schema** parameter values are:

• **first**: Analyze the schema of the first ORC, Parquet, or Avro file that **DBMS_CLOUD** finds in the specified **file_uri_list** (**first** is the default value for **schema**).

• **all**: Analyze all the schemas for all the ORC, Parquet, or Avro files found in the **file_uri_list**. Because these are simply files captured in an object store, there is no guarantee that each file's metadata is the same. For example, File1 may contain a field called "address", while File2 may be missing that field. Examining each file to derive the columns is a bit more expensive but may be required if the first file does not contain all the required fields.

---

**Note:**

If the **column_list** parameter is specified, then you provide the column names and data types for the external table and the **schema** value, if specified is ignored. Using **column_list** you can limit the columns in the external table. If **column_list** is not specified then the schema default value is **first**.

---

3. You can now run queries on the external table you created in the previous step:

```
DESC sales_extended_ext;
Name Null? Type
-------------- ----- --------------
PROD_ID NUMBER(10) 
CUST_ID NUMBER(10) 
TIME_ID VARCHAR2(4000) 
CHANNEL_ID NUMBER(10) 
PROMO_ID NUMBER(10) 
QUANTITY_SOLD NUMBER(10) 
AMOUNT_SOLD NUMBER(10,2) 
GENDER VARCHAR2(4000) 
CITY VARCHAR2(4000) 
```
This query shows values for rows in the external table. If you want to query this data frequently, after examining the data you can load it into a table with DBMS_CLOUD.COPY_DATA.

See CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files and COPY_DATA Procedure for Avro, ORC, or Parquet Files for more information.

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

Query External Tables with Partitioning Specified in Source Files

If you want to query multiple data files in the Object Store as a single external table and the files can be represented as multiple logical partitions, it is highly recommended to use an external partitioned table. Using an external partitioned table preserves the logical partitioning of your data files for query access.

Using partitioned external tables has the potential to dramatically improve query performance by only accessing the data required for the query. For example, you may have two years of daily partitions stored in separate objects on Cloud Object Store. When you use partitioned external tables, a query for a single day only needs to access that day's source data. When you use partitioned external tables the database automatically partition prunes, and in this example only needs to scan a very small fraction of the data.

There are two ways to create an external partitioned table with the DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE procedure:
See Query External Partitioned Data (with Partitioning Clause) for a description of this type of external table.

Topics

• About External Tables with Source File Partitioning
• Query External Partitioned Data with Hive Format Source File Organization
• Query External Partitioned Data with Folder Format Source File Organization
• Refresh External Partitioned Tables with Updated or Deleted Source Files

About External Tables with Source File Partitioning

On Autonomous Database you can create partitioned external tables from Hive style partitioned data or from simple folder partitioned data stored on your Cloud Object Store.

Using source file partitioning, instead of supplying a complete partition specification the procedure derives partitioning information from the file path for certain file patterns. For example, consider the following data file specifications:

• Hive style: for example: sales/country=USA/year=2020/month=01/file1.csv
• Simple folder partitioning style: for example: sales/USA/2020/01/file1.parquet

Using one of these common partitioning formats greatly simplifies both the creation and management of partitioned external tables. In addition, even though partition columns may not appear in the data file, they can still be queried using SQL. Partitioning data also improves query performance by dramatically reducing the amount of data scanned. In this example, when you query ‘USA’ data, the query can skip scanning the files for other countries.

Hive Format Partitioned Data in Cloud Object Store

Hive offers a standard metadata format for big data processing engines. Partitioned data in Cloud Object Store that is generated in Hive format is represented in a folder/subfolder format. For example, on Cloud Object Store a Hive format data file is stored as follows:

table/partition1=partition1_value/partition2=partition2_value/
data_file.csv

Files saved in Hive partitioned format provide partition information in the data file path name. The data file path name includes information about the object contents, including partition column names and partition column values (the data file does not include the partition columns and their associated values).

For example, consider an external partitioned SALES table created from Hive format data on Cloud Object Store:

.../sales/country=USA/year=2020/month=01/file1.csv
.../sales/country=USA/year=2020/month=01/file2.csv
.../sales/country=USA/year=2020/month=02/file3.csv
The Hive format partition information shows the data files in Cloud Object Store are partitioned by country, year, and month and the values for these partition columns are also specified within the Hive format path name for each data file (the path name includes values for the partitioned columns: country, year, and month).

The column names in the path will be used by the API to simplify the table definition.

Simple Folder Format Partitioned Data in Cloud Object Store

Partitioned data in Cloud Object Store that is generated in folder format is represented in a folder/subfolder format, similar to Hive format partitioned data, but the information in the path shows the column values and does not include the column names. Also, with folder format partitioned data the partition order specified in the object name is significant, and must match the order in the table columns.

For example, on Cloud Object Store a folder format data file is stored as follows:

```
table/partition1_value/partition2_value/*.parquet
```

The path includes both partition column values, in partition column order, and the data files. Autonomous Database allows you to create an external partitioned table from folder format data and you can perform a query using the specified partitions.

Files saved in folder partitioned format provide the data partition column values in the file name. Unlike Hive, the paths do not include the column name, therefore the column names must be provided. The order of partition columns is important and the order in the file name for column partition names must match the order in the `partition_columns` parameter.

About Querying Partitioned Data in Cloud Object Store

When you query external partitioned data in Hive format, the query engine understands and utilizes the partitioning information from the file path name. For example, consider an external partitioned `SALES` table where the source file, `sales/country=USA/year=2020/month=02/file3.csv` on Object Store includes the following sales data:

```
tents, 291
canoes, 22
backpacks, 378
```

The country values in the path name, and the time period values for month and year are not specified as columns within the data file. The partition column values are specified only in the path name with values shown: USA, 2020, and 02. After you create an external partitioned table with this data file you can use the partition columns and their values when you run a query on the external partitioned table.

For example:

```
SELECT year, month, product, units
FROM SALES WHERE year='2020' AND month='02' AND country='USA'
```
The benefit of creating an external partitioned table with data generated as Hive format partitioned data is that the query engine is optimized to partition prune the data to select the correct partition and the query only selects data from one partition and only needs to search a single data file. Thus, the query would only require a scan of the file3.csv file (/sales/country=USA/year=2020/month=02/file3.csv). For large amounts of data, such partition pruning can provide significant performance improvements.

Using standard Oracle Database external tables, the partition column must be available as a column within the data file to use it for queries or partition definitions. Without the special handling that is available with external partitioned tables on Autonomous Database, this would be a problem if you want to use data stored in Hive format on Cloud Object Store, as you would need to regenerate the data files to include the partition as a column in the data file.

About Creating Partitioned External Tables

When you use unstructured data stored in Hive format on Cloud Object Store and you create an external partitioned table, the columns and their types cannot be derived from the source file. Thus, the columns and their data types must be specified with the column_list parameter. To create the partitioned external tables, use the procedure DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE to specify the partition columns and their types as follows:

• The root for the file list is specified in the path name with the file_uri_list parameter. For example, http://.../sales/*
• The column names and data types are specified with the column_list parameter.
• The option partition_columns in the format parameter specifies the partition columns.
• The generated DLL includes the columns specified in the path name.

For this example, when the external table is created the country, year, and month columns are added in the column_list parameter. The external table is created with the country, year, and month columns, which are not in the data files, and list partitions are created enabling partition pruning.

When you use structured data, such as Parquet, Avro, or ORC files stored in folder format on Cloud Object Store, the columns and their data types are known, and you do not need to specify the column list as is required with unstructured data. To create the partitioned external tables, use the procedure DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE to specify the partition columns and their types as follows:

• The root for the file list is specified in the path name with the file_uri_list parameter. For example, http://.../sales/*
• The column_list parameter is not required for structured files. If you do not specify the column list, you must define the partition columns and their data types when you create the external partitioned table. Use the option partition_columns in the format parameter to specify the partition columns and their data types.
• The generated DLL includes the columns specified in the path name.

See Query External Partitioned Data with Hive Format Source File Organization and Query External Partitioned Data with Folder Format Source File Organization for complete examples.
External Partitioning: CSV Source Files with Hive-style Folders

Shows how to create external partitioned tables with CSV source files stored on Cloud Object Store in Hive-style folders.

Source file list:

```
.../sales/country=USA/year=2020/month=01/file1.csv
.../sales/country=USA/year=2020/month=01/file2.csv
.../sales/country=USA/year=2020/month=02/file3.csv
.../sales/country=USA/year=2020/month=03/file1.csv
.../sales/country=FRA/year=2020/month=03/file1.csv
```

API:

```
DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
  table_name => 'mysales',
  credential_name => 'mycredential',
  file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/.../sales/*.csv',
  column_list => 'product varchar2(100), units number, country varchar2(100), year (number), month varchar2(2)',
  field_list => 'product, units', --[Because country, year and month are not in the file, they are not listed in the field list]
  format => '{"type":"csv","partition_columns":["country", "year", "month"]}');
```

**Note:**

The `partition_columns` in the `format` parameter must match the column names found in the path (for example, the `country` column matches "country=").

External Partitioning: CSV Source Files with Simple Folders

Shows how to create external partitioned tables with CSV source files stored on Cloud Object Store in simple folder format.

Source file list:

```
.../sales/USA/2020/01/file1.csv
.../sales/USA/2020/01/file2.csv
.../sales/USA/2020/02/file3.csv
.../sales/USA/2020/03/file1.csv
.../sales/FRA/2020/03/file1.csv
```

API:

```
DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
  table_name => 'mysales',
  credential_name => 'mycredential',
```

Chapter 4

Query External Tables with Partitioning Specified in Source Files
file_uri_list => https://objectstorage.us-phoenix-1.oraclecloud.com/.../sales/*.csv,
column_list => 'product varchar2(100), units number, country varchar2(100), year (number), month varchar2(2)',
field_list => 'product, units', --[Because country, year and month are not in the file, they are not listed in the field list]
format => '"type":"csv","partition_columns":["country", "year", "month"]'};

Note:
The API call is the same as in the previous example, but the order of the partition_columns in the format parameter is significant because the column name is not in the file path.

External Partitioning: Parquet Source Files with Hive-style Folders

Shows how to create external partitioned tables with Parquet source files stored on Cloud Object Store in Hive-style folders.

Source file list:

.../sales/USA/2020/01/file1.parquet
.../sales/USA/2020/01/file2.parquet
.../sales/USA/2020/02/file3.parquet
.../sales/USA/2020/03/file1.parquet
.../sales/FRA/2020/03/file1.parquet

API:

DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE (table_name => 'mysales', credential_name => 'mycredential', file_uri_list => https://objectstorage.us-phoenix-1.oraclecloud.com/.../sales/*.parquet, format => 
  json_object(  'type' value 'parquet',  'schema' value 'first',  'partition_columns' value
    json_array(      json_object('name' value 'country', 'type' value 'varchar2(100)'),      json_object('name' value 'year', 'type' value 'number'),      json_object('name' value 'month', 'type' value 'varchar2(2)')
    )
  )
);
External Partitioning: Parquet with Simple Folders

Shows how to create external partitioned tables with Parquet source files stored on Cloud Object Store in simple folder format.

Source file list:

```plaintext
.../sales/USA/2020/01/file1.parquet
.../sales/USA/2020/01/file2.parquet
.../sales/USA/2020/02/file3.parquet
.../sales/USA/2020/03/file1.parquet
.../sales/FRA/2020/03/file1.parquet
```

API:

```sql
DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE (  
  table_name => 'mysales',  
  credential_name => 'mycredential',  
  file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/.../sales/*.parquet',  
  format =>  
    json_object(  
      'type' => 'parquet',  
      'schema' => 'first',  
      'partition_columns' =>  
        json_array(  
          json_object('name' => 'country', 'type' => 'varchar2(100)'),  
          json_object('name' => 'year', 'type' => 'number'),  
          json_object('name' => 'month', 'type' => 'varchar2(2)')  
        )  
    );
```

Note:
The `column_list` parameter is not specified. You must include both the name and data type for the partition columns. In addition, the order of the `partition_columns` in the format clause matters because the column name is not in the file path.
Query External Partitioned Data with Hive Format Source File Organization

Use `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` to create an external partitioned table and generate the partitioning information from the Cloud Object Store file path.

Consider the following sample source files in Object Store:

- custsales/month=2019-01/custsales-2019-01.csv
- custsales/month=2019-02/custsales-2019-02.csv
- custsales/month=2019-03/custsales-2019-03.csv

With this naming, the values for `month` are captured within the object name.

To create a partitioned external table with data stored in this sample Hive format, do the following:

1. **Store Object Store credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL`**.

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

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for creating external tables.

   See `CREATE_CREDENTIAL` Procedure for information about the `username` and `password` parameters for different object storage services.

2. **Create an external partitioned table on top of your source files using the procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`**.

   The procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` supports external partitioned files in the supported cloud object storage services. The credential is a table level property; therefore, the external files must all be on the same cloud object store.

   For example:
   ```sql
   BEGIN
   DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(  
     TABLE_NAME => 'sales_sample',
   END;
   ```
The parameters are:

- **table_name**: is the external table name.
- **credential_name**: is the name of the credential created in the previous step.
- **file_uri_list**: is a comma-delimited list of source file URIs. There are two options for this list:
  - Specify a comma-delimited list of individual file URIs without wildcards.
  - Specify a single file URI with wildcards, where the wildcards can only be after the last slash "/". The character "*" can be used as the wildcard for multiple characters, the character "?" can be used as the wildcard for a single character.
- **column_list**: is a comma delimited list of column names and data types for the external table. The list includes the columns inside the data file and those derived from the object name (from names in the file path).
  The **column_list** is not required when the data files are structured files (Parquet, Avro, or ORC).
- **format**: defines the options you can specify to describe the format of the source file. The **partition_columns** format parameter specifies the names of the partition columns. See DBMS_CLOUD Package Format Options for more information.

In this example, **namespace-string** is the Oracle Cloud Infrastructure object storage namespace and **bucketname** is the bucket name. See Understanding Object Storage Namespaces for more information.

The **DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE** call would result in the following table definition:

```sql
CREATE TABLE "ADMIN"."SALES_SAMPLE"
( "DAY_ID" TIMESTAMP (6),
  "GENRE_ID" NUMBER(19,0),
  "MOVIE_ID" NUMBER(19,0),
  "CUST_ID" NUMBER(19,0),
  "APP" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
  "DEVICE" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
  "OS" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
  "PAYMENT_METHOD" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
  "LIST_PRICE" BINARY_DOUBLE,
  "DISCOUNT_TYPE" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
  "DISCOUNT_PERCENT" BINARY_DOUBLE,
  "ACTUAL_PRICE" BINARY_DOUBLE,
  "MONTH" VARCHAR2(100 BYTE) COLLATE "USING_NLS_COMP"
) DEFAULT COLLATION "USING_NLS_COMP"
ORGANIZATION EXTERNAL
( TYPE ORACLE_BIGDATA
```
DEFAULT DIRECTORY "DATA_PUMP_DIR"
ACCESS PARAMETERS
{ com.oracle.bigdata.fileformat=parquet
  com.oracle.bigdata.filename.columns=["month"]
  com.oracle.bigdata.file_uri_list="https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/moviestream_landing/o/sales_sample/*.parquet"
  com.oracle.bigdata.credential.schema="ADMIN"
  com.oracle.bigdata.credential.name=CRED_OCI
  com.oracle.bigdata.trimspaces=notrim
}

REJECT LIMIT 0
PARTITION BY LIST ("MONTH")
(PARTITION "P1" VALUES ("2019-01")
 LOCATION
  ( 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/moviestream_landing/o/sales_sample/month=2019-01/*.parquet' ),
PARTITION "P2" VALUES ("2019-02")
 LOCATION
  ( 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/moviestream_landing/o/sales_sample/month=2019-02/*.parquet' )
)
PARALLEL ;

See CREATE_EXTERNAL_PART_TABLE Procedure for detailed information about the parameters.
See DBMS_CLOUD Package File URI Formats for more information on the supported cloud object storage services.

3. You can now run queries on the external partitioned table you created in the previous step.

Your Autonomous Database takes advantage of the partitioning information of your external partitioned table, ensuring that the query only accesses the relevant data files in the Object Store.

For example:

SELECT movie_id, month FROM sales WHERE month='2019-02'

The external partitioned tables you create with DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE include two invisible columns file$path and file$name. These columns help identify which file a record is coming from. See External Table Metadata Columns for more information.

If there are any rows in the source files that do not match the format options you specified, the query reports an error. You can use DBMS_CLOUD parameters, like rejectlimit, to suppress these errors. As an alternative, you can also validate the external partitioned table you created to see the error messages and the rejected rows so that you can change your format options accordingly. See Validate External Data and Validate External Partitioned Data for more information.
Query External Partitioned Data with Folder Format Source File Organization

Use `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` to create an external partitioned table and generate the partitioning information from the Cloud Object Store file path.

When you create an external table with folder format data files, you have two options for specifying the types of the partition columns:

- You can manually specify the columns and their data types with `column_list` parameter. See Query External Partitioned Data with Hive Format Source File Organization for an example using the `column_list` parameter.
- You can let `DBMS_CLOUD` derive the data file columns and their types from information in structured data files such as Avro, ORC, and Parquet data files. In this case, you use the `partition_columns` option with the `format` parameter to supply the column names and their data types for the partition columns and you do not need to supply the `column_list` or the `field_list` parameters.

Consider the following sample source files in Object Store:

```bash
.../sales/USA/2020/01/sales1.parquet
.../sales/USA/2020/02/sales2.parquet
```

To create a partitioned external table with the Cloud Object Store file path defining the partitions from files with this sample folder format, do the following:

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

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

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for creating external tables.

   See `CREATE_CREDENTIAL Procedure` for information about the `username` and `password` parameters for different object storage services.

2. Create an external partitioned table on top of your source files using the procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`. 

The procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` supports external partitioned files in the supported cloud object storage services. The credential is a table level property; therefore, the external files must all be on the same cloud object store.

For example:

```sql
BEGIN
  DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
    table_name => 'MYSALES',
    credential_name => 'DEF_CRED_NAME',
    file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
      namespace-string/b/bucketname/o/sales/*.parquet',
    format =>
      json_object('type' => 'parquet', 'schema' => 'first',
        'partition_columns' =>
          json_array(
            json_object('name' => 'country', 'type' => 'varchar2(100)'),
            json_object('name' => 'year', 'type' => 'number'),
            json_object('name' => 'month', 'type' => 'varchar2(2)')
          )
      )
  );
END;
/
```

The `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` parameters for structured data files, such as for a Parquet data file does not require the `column_list` or the `field_list` parameters. The column names and data types are derived for the columns from the first parquet file that the procedure scans (and therefore all files must have the same shape). The generated column list includes the columns derived from the object name and these column have the data types specified with the `partition_columns format` parameter.

The parameters are:

- `table_name`: is the external table name.
- `credential_name`: is the name of the credential created in the previous step.
- `file_uri_list`: is a comma-delimited list of source file URIs. There are two options for this list:
  - Specify a comma-delimited list of individual file URIs without wildcarding.
  - Specify a single file URI with wildcards, where the wildcards can only be after the last slash "/". The character "*" can be used as the wildcard for multiple characters, the character "?" can be used as the wildcard for a single character.
- `column_list`: is a comma delimited list of column names and data types for the external table. The list includes the columns that are inside the file as well as those derived from the object name.

The `column_list` is not required when the data files are structured files (Parquet, Avro, or ORC).
• **field_list**: Identifies the fields in the source files and their data types. The default value is NULL meaning the fields and their data types are determined by the `column_list` parameter.

The field_list is not required when the data files are structured files (Parquet, Avro, or ORC).

• **format**: defines the options you can specify to describe the format of the source file. The `partition_columns` format parameter specifies the names of the partition columns. See DBMS_CLOUD Package Format Options for more information.

In this example, `namespace-string` is the Oracle Cloud Infrastructure object storage namespace and `bucketname` is the bucket name. See Understanding Object Storage Namespaces for more information.

See CREATE_EXTERNAL_PART_TABLE Procedure for detailed information about the parameters.

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

If there are any rows in the source files that do not match the format options you specified, the query reports an error. You can use DBMS_CLOUD parameters, like `rejectlimit` to suppress these errors. As an alternative, you can also validate the external partitioned table you created to see the error messages and the rejected rows so that you can change your format options accordingly. See Validate External Data and Validate External Partitioned Data for more information.

3. You can now run queries on the external partitioned table you created in the previous step.

Your Autonomous Database takes advantage of the partitioning information of your external partitioned table, ensuring that the query only accesses the relevant data files in the Object Store. For example, the following query only reads data files from one partition.

For example:

```sql
SELECT year, month, product, units
FROM SALES WHERE year='2020' AND month='02' AND country='USA'
```

The external partitioned tables you create with `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from. See External Table Metadata Columns for more information.
Refresh External Partitioned Tables with Updated or Deleted Source Files

You can use `DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE` to refresh an external partitioned table. Use this procedure when new partitions are added or when partitions are removed from the object store source.

**Note:**

Only use `DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE` when the partitioned external table is created with `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` and the `file_url_path` parameter.

To refresh a partitioned external table:

1. Refresh an external partitioned table on top of your Cloud Object Store source files using the procedure `DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE`. For example:

   ```sql
   BEGIN
   DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE(table_name => 'MYSALES');
   END;
   /  
   ```

   `DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE` uses the credential information from the corresponding `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` with the specified `table_name` to access Cloud Object Store.

   See `SYNC_EXTERNAL_PART_TABLE Procedure` for detailed information about the parameter.

2. The partition information is now refreshed and you can run queries on the updated external partitioned table with new partitions available or with partitions that were removed no longer available in the external partitioned table.

Query External Partitioned Data (with Partitioning Clause)

If you want to query multiple data files in the Object Store as a single external table and the files can be represented as multiple logical partitions, then it is highly recommended to use an external partitioned table. Using an external partitioned table preserves the logical partitioning of your data files for query access. Use the procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` to create an external partitioned table.

There are two ways to create an external partitioned table on Autonomous Database:

- The first version is for. See `Query External Partitioned Data with Hive Format Source File Organization` for information on this type of external partitioned table usage.

- The second version is for. When you create a partitioned external table in this way you include a partitioning clause in the `partitioning_clause` parameter. The
partitioning clause that you include depends upon the data files in the Cloud and the type of partitioning you use. This section describes this type of external partitioned table usage.

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

   ```sql
   BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
      credential_name => 'DEF_CRED_NAME',
      username => 'adb_user@example.com',
      password => 'password');
   END;
   /
   
   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for creating external tables.

   See `CREATE_CREDENTIAL` Procedure for information about the `username` and `password` parameters for different object storage services.

2. Create an external partitioned table on top of your source files using the procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`.

   The procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` supports external partitioned files in the supported cloud object storage services. The credential is a table level property; therefore, the external files must be on the same object store.

   For example:

   ```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 Database takes advantage of the partitioning information of your external partitioned table, ensuring that the query only accesses the relevant data files in the Object Store. For example, the following query only reads data files from partition P1:

```
SELECT * FROM pet1 WHERE col1 < 750;
```

The external partitioned tables you create with `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from. See External Table Metadata Columns for more information.

If there are any rows in the source files that do not match the format options you specified, the query reports an error. You can use `DBMS_CLOUD` parameters, like `rejectlimit`, to suppress these errors. As an alternative, you can also validate the external partitioned table you created to see the error messages and the rejected rows so that you can change your format options accordingly. See Validate External Data and Validate External Partitioned Data for more information.

See CREATE_EXTERNAL_PART_TABLE Procedure for detailed information about the parameters.

See DBMS_CLOUD 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.

When you a create a hybrid partitioned table, you include a partitioning clause in the `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` statement. The partitioning clause that you
include depends upon your data files and the type of partitioning you use. See Creating Hybrid Partitioned Tables for more information.

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

   For example:

   ```sql
   BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(  
       credential_name => 'DEF_CRED_NAME',
       username => 'adb_user@example.com',
       password => 'password');
   END;
   /
   
   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for creating hybrid partitioned tables.

   See **CREATE_CREDENTIAL Procedure** for information about the `username` and `password` parameters for different object storage services.

2. Create a hybrid partitioned table on top of your source files using the procedure **DBMS_CLOUD.CREATE_HYBRID_PART_TABLE**.

   The procedure **DBMS_CLOUD.CREATE_HYBRID_PART_TABLE** supports external partitioned files in the supported cloud object storage services. The credential is a table level property; therefore, the external files must be on the same object store.

   For example:

   ```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 Database takes advantage of the partitioning information of your hybrid partitioned table, ensuring that the query only accesses relevant data files in the Object Store. For example, the following query only reads data files from partition P1:

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

The hybrid partitioned tables you create with `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from. See External Table Metadata Columns for more information.

If there are any rows in the source files that do not match the format options you specified, the query reports an error. You can use `DBMS_CLOUD` parameters, like `rejectlimit`, to suppress these errors. As an alternative, you can also validate the hybrid partitioned table you created to see the error messages and the rejected rows so that you can change your format options accordingly. See Validate External Data and Validate Hybrid Partitioned Data for more information.

See `CREATE_HYBRID_PART_TABLE Procedure` detailed information about the parameters.

See `DBMS_CLOUD 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 => 'adb_user@example.com',
       password => 'password');
   END;
   /
   ```

   This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for creating external tables.

   See `CREATE_CREDENTIAL Procedure` for information about the `username` and `password` parameters for different object storage services.

2. **Create an external table on top of your source files using the procedure** `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`.

   For example:

   ```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' value 'datapump', 'rejectlimit' value '1'),
       column_list => 'CHANNEL_ID NUMBER, CHANNEL_DESC VARCHAR2(20),
       CHANNEL_CLASS VARCHAR2(20)');
   END;
   /
   ```

   The parameters are:
   
   - **table_name**: is the external table name.
   - **credential_name**: is the name of the credential created in the previous step.
   - **file_uri_list**: is a comma delimited list of the Data Pump dump files you want to query.
   - **format**: defines the options you can specify to describe the format of the source file. 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.

### Query Big Data Service Hadoop (HDFS) Data from Autonomous Database

You can create database links to Oracle Big Data Service from Autonomous Database.

Big Data Service provides enterprise-grade Hadoop as a service, with end-to-end security, high performance, and ease of management and upgradeability. After deploying the Oracle Cloud SQL Query Server to Big Data Service, you can easily query data available on Hadoop clusters at scale from Autonomous Database using SQL. This allows you to blend data coming from your data lake with data in your Autonomous Database.

Oracle Cloud SQL Query Server is an Oracle SQL engine that can be accessed using database links in Autonomous Database. You create links to Big Data Service using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. See Integrating Cloud SQL with Autonomous Database for more information.

### Query External Data with Data Catalog

Oracle Cloud Infrastructure Data Catalog is the metadata management service for Oracle Cloud that helps you discover data and support data governance. It provides an inventory of assets, a business glossary, and a common metastore for data lakes.

Autonomous Database can leverage this metadata to dramatically simplify management for access to your data lake's object store. Instead of manually defining external tables to access your data lake, use the external tables that are defined and managed automatically. These tables will be found in Autonomous Database protected schemas that are kept up to date with changes in Data Catalog.

For more information about Data Catalog, please refer to Data Catalog documentation.
About Querying with Data Catalog

By synchronizing with Data Catalog metadata, Autonomous Database automatically creates external tables for each logical entity harvested by Data Catalog. These external tables are defined in database schemas that are fully managed by the metadata synchronization process. Users can immediately query data without having to manually derive the schema (columns and data types) for external data sources and manually create external tables.

Synchronization is dynamic, keeping the Autonomous Database up-to-date with respect to changes to the underlying data, reducing administration cost as it automatically maintains hundreds to thousands of tables. It also allows multiple Autonomous Database instances to share the same Data Catalog, further reducing management costs and providing a common set of business definitions.

The Data Catalog folders/buckets are containers that sync with Autonomous Database schemas. Logical entities within those folders/buckets map to Autonomous Database external tables. These schemas and external tables are automatically generated and maintained through the sync process:

- Folders/Buckets map to database schemas that are for organizational purposes only.
- The organization is meant to be consistent with the data lake and minimize confusion when accessing data through different paths.
- Data Catalog is the source of truth for the tables contained within schemas. Changes made in the Data Catalog update the schema’s tables during a subsequent sync.

To use this capability, a Database Data Catalog Administrator initiates a connection to a Data Catalog instance, selects which data assets and logical entities to synchronize, and runs the sync. The sync process creates schemas and external tables based on the selected Data Catalog harvested data assets and logical entities. As soon as the external tables are created, Data Analysts can start querying their data without having to manually derive the schema for external data sources and create external tables.

Note:

The DBMS_DCAT Package is available for performing the tasks required to query Data Catalog object store data assets. See DBMS_DCAT Package.

Concepts Related to Querying with Data Catalog

An understanding of the following concepts is necessary for querying with Data Catalog.
Data Catalog
Data Catalog harvests data assets that point to the object store data sources you want to query with Autonomous Database. From Data Catalog you can specify how the data is organized during harvesting, supporting different file organization patterns. As part of the Data Catalog harvesting process, you can select the buckets and files you want to manage within the asset. For further information, see Data Catalog Overview.

Object Stores
Object Stores have buckets containing a variety of objects. Some common types of objects found in these buckets include: CSV, parquet, avro, json, and ORC files. Buckets generally have a structure or a design pattern to the objects they contain. There are many different ways to structure data and many different ways of interpreting these patterns. For example, a typical design pattern uses top-level folders that represent tables. Files within a given folder share the same schema and contain data for that table. Subfolders are often used to represent table partitions (for example, a subfolder for each day). Data Catalog refers to each top-level folder as a logical entity, and this logical entity maps to an Autonomous Database external table.

Connection
A connection is an Autonomous Database connection to a Data Catalog instance. For each Autonomous Database instance there can be a single connection to a Data Catalog instance. The Autonomous Database credential must have rights to access Data Catalog assets that have harvested object storage.

Harvest
A Data Catalog process that scans object storage and generates the logical entities from your data sets.

Data Asset
A data asset in Data Catalog represents a data source, which includes databases, Oracle Object Storage, Kafka, and more. Autonomous Database leverages Oracle Object Storage assets for metadata synchronization.

Data Entity
A data entity in Data Catalog is a collection of data such as a database table or view, or a single file and normally has many attributes that describe its data.

Logical Entity
In Data Lakes, numerous files typically comprise a single logical entity. For example, you may have daily clickstream files, and these files share the same schema and file type. A Data Catalog logical entity is a group of Object Storage files that are derived during harvesting by applying filename patterns that have been created and assigned to a data asset.

Data Object
A data object in Data Catalog refers to data assets and data entities.

Filename Pattern
In a data lake, data may be organized in different ways. Typically, folders capture files of the same schema and type. You must register to Data Catalog how your data is organized. Filename patterns are used to identify how your data is organized. In Data Catalog, you can define filename patterns using regular expressions. When Data
Catalog harvests a data asset with an assigned filename pattern, logical entities are created based on the filename pattern. By defining and assigning these patterns to data assets, multiple files can be grouped as logical entities based on the filename pattern.

Synchronize (Sync)
Autonomous Database performs synchronizations with Data Catalog to automatically keep its database up-to-date with respect to changes to the underlying data. Synchronization can be performed manually, or on a schedule. The sync process creates schemas and external tables based on the Data Catalog data assets and logical entities. These schemas are protected, which means their metadata is managed by Data Catalog. If you want to alter the metadata, you must make the changes in Data Catalog. The Autonomous Database schemas will reflect any changes after the next sync is run. For further details, see Synchronization Mapping.

Synchronization Mapping

The synchronization process creates and updates Autonomous Database schemas and external tables based on Data Catalog data assets, folders, logical entities, attributes and relevant custom overrides.
### Data Catalog and Autonomous Database Mapping Description

#### Default values:

By default, the generated schema name in Autonomous Database has the following format:

DCAT$<data-asset-name>_<folder-name>

- data-asset-name is the name of the Data Catalog data asset's name.
- folder-name is the Data Catalog folder name. This folder maps to an object storage bucket.

#### Customizations:

The default data-asset-name and folder-name can be customized by defining custom properties, business names and display names to override these default names.

- data-asset-name can be overridden by defining the oracle-db-schema-prefix custom property for the data asset in Data Catalog.
- folder-name can be overridden by defining the oracle-db-schema custom property for the folder in Data Catalog, a business name or display name. The following attributes are used in order of precedence for generating the folder-name:
  1. oracle-db-schema custom property
  2. Business Name
  3. Display Name

#### Examples:

- If the data asset name is MYASSET, the folder name is MYFOLDER, and there are no custom property overrides, the schema name is: DCAT$MYASSET_MYFOLDER
- If the data asset has oracle-db-prefix = FIRSTASSET, and the folder has oracle-db-schema = FIRSTFOLDER, then the schema name is: DCAT$FIRSTASSET_FIRSTFOLDER

### Logical entity

Logical entities are mapped to external tables. If the logical entity has a partitioned attribute, it is mapped to a partitioned external table.

- The external table name is derived from the corresponding logical entity's Display Name, or Business Name.
- If oracle-db-schema is set, then its value overrides all the names and custom properties of the corresponding folders and data assets.
- For example, if oracle-db-schema for an entity is set to EntitySchema, then the table is created in schema DCAT$ENTITYSHEMA.
Data Catalog | Autonomous Database | Mapping Description
---|---|---
Logical entity’s attributes | External table columns | **Column names:** The external table column names are derived from the corresponding logical entity’s attribute display names, or business names.

For logical entities derived from Parquet, Avro, and ORC files, the column name is always the display name of the attribute as it represents the field name derived from the source files.

For attributes corresponding to a logical entity derived from CSV files, the following attribute fields are used in order of precedence for generating the column name:

1. `oracle-db-column-name`
2. Business Name
3. Display Name

**Column type:** The `oracle-db-column-type` custom property overrides the default column type that was derived by Data Catalog.

For attributes corresponding to a logical entity derived from Avro files with `TIME_MICROS`, `TIME_MILLIS`, `TIMESTAMP_MICROS` or `TIMESTAMP_MILLIS` data types, you must set the `oracle-db-column-type` of the corresponding attribute in Data Catalog.

**Column length:** The `oracle-db-column-length` custom property overrides the default column length for a string field that was derived by Data Catalog.

**Column precision:** The `oracle-db-column-precision` custom property overrides the default precision for a number that was derived by Data Catalog.

For attributes corresponding to a logical entity derived from Avro files with `TIME_MICROS`, `TIME_MILLIS`, `TIMESTAMP_MICROS` or `TIMESTAMP_MILLIS` data types, you must set the `oracle-db-column-precision` of the corresponding attribute in Data Catalog.

**Column scale:** The `oracle-db-column-scale` custom property overrides the default scale for a number that was derived by Data Catalog.

---

**Typical Workflow with Data Catalog**

There is a typical workflow of actions performed by users who want to query with Data Catalog.

The Database Data Catalog Admin creates a connection between the Autonomous Database instance and a Data Catalog instance, then configures and runs a synchronization (sync) between the Data Catalog and Autonomous Database. The sync creates external tables and schemas in the Autonomous Database instance based on the synced Data Catalog contents. The Database Data Catalog Query Admin or Database Admin grants READ access to the generated external tables so that Data Analysts and other database users can browse and query the external tables.
The table below describes each action in detail. For a description of the different user types included in this table, see Data Catalog Users and Roles.

<table>
<thead>
<tr>
<th>Action</th>
<th>Who is the user</th>
<th>Description</th>
</tr>
</thead>
</table>
| Create policies             | Database Data Catalog Administrator | The Autonomous Database resource principal or Autonomous Database user credential must have the appropriate permissions to manage Data Catalog and to read from object storage.  
More information: Required Credentials and IAM Policies.  

| Create credentials          | Database Data Catalog Administrator | Ensure database credentials are in place to access Data Catalog and to query object store. The user calls DBMS_CLOUD.CREATE_CREDENTIAL to create user credentials and/or DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL to enable resource principals.  
More information: DBMS_CLOUD CREATE_CREDENTIAL Procedure, Use Resource Principal with DBMS_CLOUD. |
| Create connections to Data Catalog | Database Data Catalog Administrator | To initiate a connection between an Autonomous Database instance and a Data Catalog instance the user calls DBMS_DCAT.SET_DATA_CATALOG_CONN to specify a target Data Catalog instance.  
The connection to the Data Catalog instance must use a database credential object with sufficient Oracle Cloud Infrastructure (OCI) privileges. For example, the Resource Principal Service Token for the Autonomous Database instance or an OCI user with sufficient privileges can be used.  
Once the connection has been made, the Data Catalog instance is updated with the DBMS_DCAT namespace and custom properties (if they do not already exist). The user can run a query to see the new connection including all current connections:  

select * from all_dcat_connections;  


Note:  
The DBMS_DCAT Package is available for performing the tasks required to query Data Catalog object store data assets. See DBMS_DCAT Package.
<table>
<thead>
<tr>
<th>Action</th>
<th>Who is the user</th>
<th>Description</th>
</tr>
</thead>
</table>
| Create a selective sync         | Database Data Catalog Administrator      | Create a sync job by selecting the Data Catalog objects to sync. The user can:  
  • Select data assets/folders to sync.  
  • Select individual logical entities to sync.  
  • Preview the resulting external tables before syncing.  
  • Change external tables (for example, the name) by modifying custom properties on Data Catalog.  
  More information: See CREATE_SYNC_JOB Procedure, DROP_SYNC_JOB Procedure, Synchronization Mapping  

Sync with Data Catalog            | Database Data Catalog Administrator      | The user initiates a sync operation. The sync is initiated manually through the DBMS_DCAT.RUN_SYNC procedure call, or automatically as part of a scheduled sync job. The sync operation creates, modifies and drops external tables and schemas according to the Data Catalog contents and sync selections. Manual configuration is applied using Data Catalog Custom Properties.  
  More information: See RUN_SYNC Procedure, CREATE_SYNC_JOB Procedure, Synchronization Mapping  

Monitor sync and view logs       | Database Data Catalog Administrator      | The user can view the sync status by querying the USER_LOAD_OPERATIONS view. After the sync process has completed, the user can view a log of the sync results, including details about the mappings of logical entities to external tables.  
  More information: Monitoring and Troubleshooting Loads  

Grant privileges                 | Database Data Catalog Query Administrator, Database Administrator | The database Data Catalog Query Administrator or database Administrator must grant READ on generated external tables to data analyst users. This allows the data analysts to query the generated external tables.  

Browse and query external tables | Data Analyst                           | Data analysts are able to query the external tables through any tool or application that supports Oracle SQL.  
  Data Analysts can review the synced schemas and tables in the DCAT$* schemas, and query the tables using Oracle SQL.  
  More information: Synchronization Mapping  

Terminate connections to Data Catalog | Database Data Catalog Administrator | To remove an existing Data Catalog association, the user calls the UNSET_DATA_CATALOG_CONN procedure.  
  This action is only done when you no longer plan on using Data Catalog and the external tables that are derived from the catalog. This action deletes Data Catalog metadata, and drops synced external tables from the Autonomous Database instance. The custom properties on Data Catalog and OCI policies are not affected.  
  More information: UNSET_DATA_CATALOG_CONN Procedure |
Example: MovieStream Scenario

In this scenario, Moviestream is capturing data in a landing zone on object storage. Much of this data, but not necessarily all, is then used to feed an Autonomous Database. Prior to feeding Autonomous Database, the data is transformed, cleansed and subsequently stored in the "gold" area.

Data Catalog is used to harvest these sources and then provide a business context to the data. Data Catalog metadata is shared with Autonomous Database, allowing Autonomous Database users to query those data sources using Oracle SQL. This data may be loaded into Autonomous Database or queried dynamically using external tables.

For more information on using Data Catalog, see Data Catalog Documentation.

1. Object Store - Review buckets, folders and files
   a. Review the buckets in your object store.
      For example, below are the landing (moviestream_landing) and gold zone (moviestream_gold) buckets in object storage:

   ![Bucket Listing](image1)

   b. Review the folders and files in the object store buckets.
      For example, below are the folders in the landing bucket (moviestream_landing) in object storage:

   ![Folder Listing](image2)

2. Data Catalog - Create filename patterns
   a. Inform Data Catalog how your data is organized using filename patterns. These are regular expressions used to categorize files. The filename patterns are used by the Data Catalog harvester to derive logical entities. The following two filename patterns are used to harvest the buckets in the MovieStream example. See Harvesting Object Storage Files as Logical Data Entities for further details on creating filename patterns.
b. To create filename patterns, go to the Filename Patterns tab for your Data Catalog and click Create Filename Pattern. For example, the following is the Create Filename Pattern tab for the moviestream Data Catalog:

![Create Filename Pattern](image)

3. Data Catalog - Data Asset Creation

a. Create a data asset that is used to harvest data from your object store. For example, a data asset named phoenixObjStore is created in the moviestream Data Catalog:

![Create Data Asset](image)

b. Add a connection to your data asset.
In this example, the data asset connects to the compartment for the moviestream object storage resource.

c. Now, associate your filename patterns with your data asset. Select Assign Filename Patterns, check the patterns you want and click Assign.

For example, here are the patterns assigned to the phoenixObjStore data asset:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hive</td>
<td>Pattern used to capture...</td>
<td>{bucketName:.+)/logic...</td>
</tr>
<tr>
<td>Single CSV File</td>
<td>A single CSV file representation</td>
<td>{bucketName: [S]+)/[o...</td>
</tr>
<tr>
<td>Folder-based Logical</td>
<td>Each folder off the root...</td>
<td>{bucketName: [w]+)/[o...</td>
</tr>
</tbody>
</table>

Assign filename patterns.
4. Data Catalog - Harvest data from object store
   
a. Harvest the Data Catalog data asset. Select the object store buckets containing the source data.
   
   In this example, the moviestream_gold and moviestream_landing buckets from object store are selected for harvesting.

   ![Harvest: phoenixObjStore](image1)

   b. After running the job, you see the logical entities. Use the Browse Data Assets to review them.

   In this example, you are looking at the customer-extension logical entity and its attributes.

   ![Logical Entity: customer-extension](image2)

   If you have a glossary, Data Catalog recommends categories and terms to associate with the entity and its attributes. This provides a business context for the items. Schemas, tables and columns are oftentimes not self-explanatory.
In our example, we want to differentiate between the different types of buckets and the meaning of their content:

- what is a landing zone?
- how accurate is the data?
- when was it last updated?
- what is the definition of a logical entity or its attribute

5. Autonomous Database - Connect to Data Catalog

Connect Autonomous Database to Data Catalog. You need to ensure that the credential used to make that connection is using an OCI principal that is authorized to access the Data Catalog asset. For further information, see Data Catalog Policies and Access Cloud Resources by Configuring Policies and Roles.

a. Connect to Data Catalog

```sql
-- Variables are used to simplify usage later
define oci_credential = 'OCI$RESOURCE_PRINCIPAL'
define dcat_ocid = 'ocid1.datacatalog.oc1.iad.aaaaaaardp66bg....twiq'
define dcat_region='us-ashburn-1'
define uri_root = 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/mytenancy/b/landing/o'
define uri_private = 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/mytenancy/b/private_data/o'

-- Run as admin
-------
-- Enable resource principal support
-------
exec dbms_cloud_admin.enable_resource_principal();

-- Test to make sure credential was created. Returns a row if it was successful
select *
from dba_credentials
where credential_name = 'OCI$RESOURCE_PRINCIPAL' and owner = 'ADMIN';

-- Query a private bucket to test the principal and privileges.
select *
from dbms_cloud.list_objects('&oci_credential', '&uri_private/');

------
-- Set the credentials to use for object store and data catalog
-- Connect to Data Catalog
-- Review connection
------
-- Set credentials
exec dbms_dcat.set_data_catalog_credential(credential_name => '&oci_credential');
exec dbms_dcat.set_object_store_credential(credential_name => '&oci_credential');
```
-- Connect to Data Catalog
begin
    dbms_dcat.set_data_catalog_conn {
        region => '&dcat_region',
        catalog_id => '&dcat_ocid');
end;
/
-- Review the connection
select * from all_dcat_connections;

b. Sync Data Catalog with Autonomous Database. Here, we'll sync all the object storage assets:

-- Sync Data Catalog with Autonomous Database
---- Let's sync all of the assets.
begin
    dbms_dcat.run_sync('{"asset_list":[]}');
end;
/

-- View log
select type, start_time, status, logfile_table from user_load_operations; -- Logfile_Table will have the name of the table containing the full log.
select * from dbms_dcat$1_log;

-- View the new external tables
select * from dcat_entities;
select * from dcat_attributes;

c. Autonomous Database - Now start running queries against object store.

-- Query the Data!
select * from dcat$phoenixobjstore_moviestream_gold.genre;

6. Change schemas for objects
The default schema names are rather complicated. Let's simplify them by specifying both the asset and the folder's Oracle-Db-Schema custom attribute in Data Catalog. Change the data asset to PHX and the folders to landing and gold respectively. The schema is a concatenation of the two.

a. From Data Catalog, navigate to the moviestream_landing bucket and change the asset to landing and gold respectively.

Before change:
After change:

b. Run another sync.

Example: Partitioned Data Scenario

This scenario illustrates how to create external tables in Autonomous Database that are based on Data Catalog logical entities harvested from partitioned data in Object Store.

The following example is based on Example: MovieStream Scenario and has been adapted to demonstrate integrating with partitioned data. Data Catalog is used to harvest these sources and then provide a business context to the data. For further details about this example, see Example: MovieStream Scenario.

For more information on using Data Catalog, see Data Catalog Documentation.

1. Object Store - Review buckets, folders and files
   a. Review the buckets in your object store.
For example, below are the landing (moviestream_landing) and gold zone (moviestream_gold) buckets in object storage:

b. Review the folders and files in the object store buckets.

For example, below are the folders in the landing bucket (moviestream_landing) in object storage:

2. Data Catalog - Create filename patterns

a. Inform Data Catalog how your data is organized using filename patterns. These are folder prefixes or regular expressions used to categorize files. The filename patterns are used by the Data Catalog harvester to derive logical entities. When a folder prefix is specified, the Data Catalog automatically generates logical entities from the specified folder prefix in the object store. The following filename pattern is used to harvest the buckets in the MovieStream example. See Harvesting Object Storage Files as Logical Data Entities for further details on creating filename patterns.

<table>
<thead>
<tr>
<th>Folder prefix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workshop.db/</td>
<td>Creates logical entities for sources that contain &quot;workshop.db&quot; path in the object store.</td>
</tr>
</tbody>
</table>
b. To create filename patterns, go to the **Filename Patterns** tab for your Data Catalog and click **Create Filename Pattern**. For example, the following is the Create Filename Pattern tab for the moviestream Data Catalog:

![Create Filename Pattern](image)

3. **Data Catalog - Data Asset Creation**
   
a. Create a data asset that is used to harvest data from your object store.

   For example, a data asset named `amsterdamObjStore` is created in the moviestream Data Catalog:

   ![Create Data Asset](image)

b. Add a connection to your data asset.
In this example, the data asset connects to the compartment for the moviestream object storage resource.

![Image of Oracle Cloud interface](image)

**c.** Now, associate your filename patterns with your data asset. Select **Assign Filename Patterns**, check the patterns you want and click **Assign**.

For example, here are the patterns assigned to the amsterdamObjStore data asset:

![Image of Assign Filename Patterns](image)

**4. Data Catalog - Harvest data from object store**

**a.** Harvest the Data Catalog data asset. Select the object store buckets containing the source data.
In this example, the `moviestream_gold` and `moviestream_landing` buckets from object store are selected for harvesting.

b. After running the job, you see the logical entities. Use the **Browse Data Assets** to review them.

In this example, you are looking at the `sales_sample_parquet` logical entity and its attributes. Note that Data Catalog has identified the `month` attribute as partitioned.

5. **Autonomous Database - Connect to Data Catalog**

Connect Autonomous Database to Data Catalog. You need to ensure that the credential used to make that connection is using an OCI principal that is authorized to access the Data Catalog asset. For further information, see [Data Catalog Policies](#) and [Access Cloud Resources by Configuring Policies and Roles](#).

a. **Connect to Data Catalog**

```bash
-- Variables are used to simplify usage later
define oci_credential = 'OCI$RESOURCE_PRINCIPAL'
define dcat_ocid = 'ocid1.datacatalog.oc1.eu-amsterdam-1....leguurn3dmqa'
define dcat_region='eu-amsterdam-1'
define uri_root = 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/mytenancy/b/landing/o'
define uri_private = 'https://objectstorage.us-
```
ashburn-1.oraclecloud.com/n/mytenancy/b/private_data/o'

-- Run as admin
-------
-- Enable resource principal support
-------
exec dbms_cloud_admin.enable_resource_principal();

-- Test to make sure credential was created. Returns a row if it was successful
select *
from dba_credentials
where credential_name = 'OCI$RESOURCE_PRINCIPAL' and owner = 'ADMIN';

-- Query a private bucket to test the principal and privileges.
select *
from dbms_cloud.list_objects('&oci_credential', '&uri_private/');

-------
-- Set the credentials to use for object store and data catalog
-- Connect to Data Catalog
-- Review connection
-------
-- Set credentials
exec dbms_dcat.set_data_catalog_credential(credential_name => '
&oci_credential');
exec dbms_dcat.set_object_store_credential(credential_name => '
&oci_credential');

-- Connect to Data Catalog
begin
    dbms_dcat.set_data_catalog_conn {
        region => '&dcat_region',
        catalog_id => '&dcat_ocid'});
end;
/
-- Review the connection
select * from all_dcat_connections;

b. Sync Data Catalog with Autonomous Database. Here, we'll sync all the object storage assets:

-- Sync Data Catalog with Autonomous Database
---- Let's sync all of the assets.
begin
    dbms_dcat.run_sync('"asset_list":[]');
end;
/

-- View log
select type, start_time, status, logfile_table from
user_load_operations; -- Logfile_Table will have the name of the
table containing the full log.
select * from dbms_dcat$x_log;
-- View the new external tables
select * from dcat_entities;
select * from dcat_attributes;

c. Autonomous Database - Now start running queries against object store.

-- Query the Data!
select count(*) from DCASTAMERDAMOBJSTORE_MOVISTREAM_LANDING.SALES_SAMPLE_PARQUET;

-- Examine the generated partitioned table
select dbms_metadata.get_ddl('TABLE','SALES_SAMPLE_PARQUET','DCASTAMERDAMOBJSTORE_MOVISTREAM_LANDING') from dual;

CREATE TABLE "DCASTAMERDAMOBJSTORE_MOVISTREAM_LANDING"."SALES_SAMPLE_PARQUET"
(   "MONTH" VARCHAR2(4000) COLLATE "USING_NLS_COMP",
   "DAY_ID" TIMESTAMP (6),
   "GENRE_ID" NUMBER(20,0),
   "MOVIE_ID" NUMBER(20,0),
   "CUST_ID" NUMBER(20,0),
   ...
)  DEFAULT COLLATION "USING_NLS_COMP"
ORGANIZATION EXTERNAL
( TYPE ORACLE_BIGDATA
  ACCESS PARAMETERS
   ( com.oracle.bigdata.fileformat=parquet
      com.oracle.bigdata.filename.columns=["MONTH"]
      ...
   )
)
REJECT LIMIT 0
PARTITION BY LIST ("MONTH")
(PARTITION "P1" VALUES (('2019-01'))
PARTITION "P2" VALUES (('2019-02'))
...PARTITION "P24" VALUES (('2019-01'))
 LOCATION ( 'https://swiftobjectstorage.eu-amsterdam-1.oraclecloud.com/v1/tenancy/moviestream_landing/workshop.db/sales_sample_parquet/month=2020-12/*'))
PARALLEL

6. Change schemas for objects
The default schema names are rather complicated. Let's simplify them by specifying both the asset and the folder's Oracle-Db-Schema custom attribute in Data Catalog. Change the data asset to PHX and the folders to landing and gold respectively. The schema is a concatenation of the two.

a. From Data Catalog, navigate to the moviestream_landing bucket and change the asset to landing and gold respectively.

Before change:

![Before change image]

After change:

![After change image]

b. Run another sync.

Validate External Data

To validate any external table, you can use the procedure DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE.

To validate a partitioned external table, see Validate External Partitioned Data. This procedure includes a parameter that lets you specify a specific partition to validate.

To validate a hybrid partitioned table, see Validate Hybrid Partitioned Data. This procedure includes a parameter that lets you specify a specific partition to validate.
Before validating an external table you need to create the external table. To create an external table use the procedure for your table type, either DBMS_CLOUD.CREATE_EXTERNAL_TABLE. For example:

```
BEGIN
  DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE (table_name => 'CHANNELS_EXT');
END;
/
```

This procedure scans your source files and validates them using the format options specified when you create the external table.

The validate operation, by default, scans all the rows in your source files and stops when a row is rejected. If you want to validate only a subset of the rows, use the rowcount parameter. When the rowcount parameter is set the validate operation scans rows and stops either when a row is rejected or when the specified number of rows are validated without errors.

For example, the following validate operation scans 100 rows and stops when a row is rejected or when 100 rows are validated without errors:

```
BEGIN
  DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE (table_name => 'CHANNELS_EXT', rowcount => 100);
END;
/
```

If you do not want the validate to stop when a row is rejected and you want to see all rejected rows, set the stop_on_error parameter to FALSE. In this case VALIDATE_EXTERNAL_TABLE scans all rows and reports all rejected rows.

If you want to validate only a subset of rows use the rowcount parameter. When rowcount is set and stop_on_error is set to FALSE, the validate operation scans rows and stops either when the specified number of rows are rejected or when the specified number of rows are validated without errors. For example, the following example scans 100 rows and stops when 100 rows are rejected or when 100 rows are validated without errors:

```
BEGIN
  DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE (table_name => 'CHANNELS_EXT', rowcount => 100, stop_on_error => FALSE);
END;
/
```

See VALIDATE_EXTERNAL_TABLE Procedure for detailed information about DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE parameters.

See View Logs for Data Validation to see the results of validate operations in the tables dba_load_operations and user_load_operations.
Validate External Partitioned Data

To validate an external partitioned table, you can use the procedure DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE. This procedure includes a parameter that lets you specify a specific partition to validate.

Before validating an external partitioned table you need to create the external partitioned table. To create an external partitioned table use the procedure DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE (see Query External Partitioned Data (with Partitioning Clause) for more details):

```sql
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:

```sql
BEGIN
    DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE (table_name => 'PET1', rowcount => 100);
END;
/
```

If you do not want the validate to stop when a row is rejected and you want to see all rejected rows, set the stop_on_error parameter to FALSE. In this case DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE scans all rows and reports all rejected rows.

If you want to validate only a subset of rows use the rowcount parameter. When rowcount is set and stop_on_error is set to FALSE, the validate operation scans rows and stops either when the specified number of rows are rejected or when the specified number of rows are validated without errors. For example, the following example scans 100 rows and stops when 100 rows are rejected or when 100 rows are validated without errors:

```sql
BEGIN
    DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE (table_name => 'PET1',
        rowcount => 100, stop_on_error => FALSE);
END;
/
```
Validate Hybrid Partitioned Data

To validate a hybrid partitioned table, you can use the procedure DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE. This procedure includes a parameter that lets you specify a specific partition to validate.

Before validating a hybrid partitioned table you need to create the table. To create a hybrid partitioned table use the procedure DBMS_CLOUD.CREATE_HYBRID_PART_TABLE (see Query Hybrid Partitioned Data for more details):

```sql
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:

```
BEGIN
    DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE (
        table_name => 'HPT1',
        rowcount => 100,
        stop_on_error => FALSE
    );
END;
/
```

For detailed information about DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE parameters see VALIDATE_HYBRID_PART_TABLE Procedure.

See View Logs for Data Validation to see the results of validate operations in the tables dba_load_operations and user_load_operations.

View Logs for Data Validation

To validate an external table, use the procedures DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE, DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE, and DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE.

After you validate your source files you can see the result of the validate operation by querying a load operations table:

- dba_load_operations: shows all validate operations.
- user_load_operations: shows the validate operations in your schema.

You can use these files to view load validation information. For example use this select operation to query user_load_operations:

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

<table>
<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>-----------</td>
<td>------</td>
<td>--------</td>
<td>------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>BADFILE_TABLE</td>
<td>-----------</td>
<td>------</td>
<td>--------</td>
<td>------------------</td>
<td>-------------------</td>
</tr>
</tbody>
</table>

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:

```sql
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:

```sql
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 Database

Autonomous Database includes support for building dashboards, reports, and notebooks for data analysis using Oracle Analytics Desktop and Oracle Machine Learning Notebooks.

Oracle Analytics Desktop is designed for business users and report developers. Oracle Analytics Desktop enables you to gain insight into your data and create reports and build interactive visualizations. It also gives you a preview of the self-service visualization capabilities included in Oracle Analytics Cloud, which extends the data exploration and visualization experience by offering secure sharing and collaboration across the enterprise, additional data sources, and a full mobile experience including proactive self-learning analytics delivered to your device.

Oracle Machine Learning Notebooks is a built-in browser-based interactive data analysis environment that is part of Autonomous Database. Oracle Machine Learning Notebooks is designed for data scientists who want use the extensive library of in-database machine learning features.

Topics

- Create Dashboards and Reports to Analyze and Visualize Your Data
- Create Notebooks, Workspaces, and Projects with Oracle Machine Learning Notebooks

Create Dashboards and Reports to Analyze and Visualize Your Data

Topics

- Use Oracle Analytics Desktop with Autonomous Database
- Use Oracle Analytics Cloud with Autonomous Database

Use Oracle Analytics Desktop with Autonomous Database

Gain insight into your data with Oracle Analytics Desktop. Oracle Analytics Desktop lets you explore your Autonomous Database data through interactive visualizations.

Oracle Analytics Desktop provides powerful personal data exploration and visualization in a simple per-user desktop download. Oracle Analytics Desktop is the perfect tool for quick exploration of sample data from multiple sources or for rapid analysis and investigation of your own local data sets.

Oracle Analytics Desktop makes it easy to visualize your Autonomous Database data so you can focus on exploring interesting data patterns. Just connect to Autonomous Database, select the elements that you’re interested in, and let Oracle Analytics Desktop find the best way to visualize it. Choose from a variety of visualizations to look at data in a specific way.
Oracle Analytics Desktop also gives you a preview of the self-service visualization capabilities included in Oracle Analytics Cloud, Oracle’s industrial-strength cloud analytics platform. Oracle Analytics Cloud extends the data exploration and visualization experience by offering secure sharing and collaboration across the enterprise, additional data sources, greater scale, and a full mobile experience including proactive self-learning analytics delivered to your device. Try Oracle Analytics Desktop for personal analytics and to sample a taste of Oracle’s broader analytics portfolio.

Oracle Analytics Desktop’s benefits include:

- A personal, single-user desktop application.
- Offline availability.
- Completely private analysis.
- Full control of data source connections.
- Lightweight single-file download.
- No remote server infrastructure.
- No administration tasks.

See the *User’s Guide for Oracle Analytics Desktop* for information on connecting Oracle Analytics Desktop to Autonomous Database.

To get started with Oracle Analytics Desktop use the following link to visit the software download page which includes more information about system requirements and provides instructions for installing Oracle Analytics Desktop on different platforms:

*Oracle Analytics Desktop Download Details*

### Use Oracle Analytics Cloud with Autonomous Database

You can use Oracle Analytics Cloud with Autonomous Database. Oracle Analytics Cloud provides a complete set of tools for deriving and sharing data insights.

- Data preparation: Analysts can ingest, profile, and cleanse data using a variety of algorithms.
- Data flow: Analysts can prepare, transform and aggregate data, and then run machine-learning models at scale.
- Data discovery: Subject matter experts can easily collaborate with other business users, blending intelligent analysis at scale, machine learning, and statistical modeling.
- Data visualization: Analysts can visualize any data, on any device, on-premise and in the cloud.
- Data collaboration: Large organizations and small teams can share data more simply, as you don’t need to manage or consolidate multiple versions of spreadsheets.
- Data-driven: Application developers can utilize interfaces that enable them to extend, customize, and embed rich analytic experiences in the application flow.

See *Visualizing Data and Building Reports in Oracle Analytics Cloud* for details on connecting Autonomous Database with Oracle Analytics Cloud.
Create Notebooks, Workspaces, and Projects with Oracle Machine Learning Notebooks

Oracle Machine Learning Notebooks provides a collaborative interface for creating notebooks and for performing real-time analytics.

Topics

- Using Oracle Machine Learning Notebooks on Autonomous Database
- Work with Oracle Machine Learning Notebooks for Data Access, Analysis, and Discovery

Using Oracle Machine Learning Notebooks on Autonomous Database

Oracle Machine Learning Notebooks provide a browser-based interactive data analysis environment where you can develop, document, share, and automate analytical methodologies.

Oracle Machine Learning Notebooks includes:

- **Oracle Machine Learning User Administration Application**
  - Web based administrative UI for managing (list, create, update, delete) Oracle Machine Learning users
  - Notebook users map to database users.
  - Access to the User Management feature is limited to the database administrator account (ADMIN). See Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database for more information.

- **Oracle Machine Learning Notebooks Application**
  - Web based application for data scientists
  - Allows for creation of workspaces, projects, and notebooks

  See Work with Oracle Machine Learning Notebooks for Data Access, Analysis, and Discovery for information on accessing Oracle Machine Learning user workspaces, projects, and notebooks from your Autonomous Database.

Work with Oracle Machine Learning Notebooks for Data Access, Analysis, and Discovery

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

To access Oracle Machine Learning Notebooks you can use the Oracle Cloud Infrastructure Console or Database Actions.

To access Oracle Machine Learning Notebooks from the Oracle Cloud Infrastructure Console:

1. From the Display Name column, select an Autonomous Database.
2. On the Autonomous Database Details page click the **Tools** tab.
3. In the **Oracle ML User Administration** area, click **Open Oracle ML User Administration**.

4. Enter your **Username** and **Password**.
   The Administrator creates OML users. See Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database for details.

5. Click **Sign in**.

To access Oracle Machine Learning Notebooks from Database Actions:

1. On the Oracle Cloud Infrastructure Console, from the Display Name column select an Autonomous Database.

2. On the Autonomous Database Details page click **Database Actions**.

3. On the Database Actions Launchpad, under **Development**, select **Oracle Machine Learning**.

4. Enter your username and password.
   The Administrator creates OML users. See Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database for details.

5. Click **Sign In**.

This shows Oracle Machine Learning user application.

Oracle Machine Learning Notebooks allows you to access your data in your database and build notebooks with the following:

- Data Ingestion and Selection
- Data Viewing and Discovery
- Data Graphing, Visualization, and Collaboration
- Data Analysis

You can also create and run SQL statements and create and run SQL scripts that access your data in your database.
To export data from an Autonomous Database, use one of the following methods:

- Use Oracle Data Pump to export the data to a directory on your database, and then move the data from the directory to Cloud Object Storage.
- Use Oracle Data Pump to export the data to Cloud Object Storage directly. This method is only supported with Oracle Cloud Infrastructure Object Storage and Oracle Cloud Infrastructure Object Storage Classic.
- Use the procedure `DBMS_CLOUD.EXPORT_DATA` to export data to your Cloud Object Store as Oracle Data Pump dump files, by specifying a query. This method is only supported with Oracle Cloud Infrastructure Object Storage and Oracle Cloud Infrastructure Object Storage Classic.
- Use the procedure `DBMS_CLOUD.EXPORT_DATA` to export data to your Cloud Object Store as text, by specifying a query. This export method supports exporting data as CSV, JSON, or XML files.

This export method supports all the Cloud Object Stores supported by Autonomous Database, and you can use an Oracle Cloud Infrastructure resource principal to access your Oracle Cloud Infrastructure Object Store or Amazon Resource Names (ARNs) to access AWS Simple Storage Service (S3).
- Use Oracle GoldenGate Capture for Oracle Autonomous Database.

Topics

- Move Data with Data Pump Export to an Autonomous Database Directory
- Move Data with Data Pump Export to Object Store
- Move Data to Object Store as Oracle Data Pump Files Using EXPORT_DATA
- Download Dump Files, Run Data Pump Import, and Clean Up Object Store
- Access Log Files for Data Pump Export
- Move Data to Object Store as CSV, JSON, or XML Using EXPORT_DATA
- Oracle GoldenGate Capture for Oracle Autonomous Database

Move Data with Data Pump Export to an Autonomous Database Directory

Oracle Data Pump offers very fast bulk data and metadata movement between Autonomous Database and other Oracle databases. To move data from an Autonomous Database to another Oracle database, use Oracle Data Pump to export to a directory on your Autonomous Database.
This export method is supported on all of the object stores that Autonomous Database supports:

- Oracle Cloud Infrastructure Object Storage
- Oracle Cloud Infrastructure Object Storage Classic
- Azure Blob Storage
- Amazon S3

**Note:**

If you are using Oracle Cloud Infrastructure Object Storage or Oracle Cloud Infrastructure Object Storage Classic, then you can use the alternative methods to export directly to object store. See Move Data with Data Pump Export to Object Store and Move Data to Object Store as Oracle Data Pump Files Using EXPORT_DATA for more information.

This export method includes the following steps:

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

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

3. Download the dump file set from the cloud object store, run Data Pump Import, and then perform any required clean up such as removing the dump file set from your cloud object store.
   See Download Dump Files, Run Data Pump Import, and Clean Up Object Store for details.

### Use Data Pump to Create a Dump File Set on Autonomous Database

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

Oracle recommends using the latest Oracle Data Pump version for exporting data from Autonomous Database to other Oracle databases, as it contains enhancements and fixes for a better experience. Download the latest version of Oracle Instant Client and download the Tools Package, which includes Oracle Data Pump, for your platform. See the installation instructions on the platform install download page for the installation steps required after you download Oracle Instant Client and the Tools Package. See Oracle Instant Client Downloads for details.
1. Run Data Pump Export with the `dumpfile` parameter set, the `filesize` parameter set to less than 5G, and the `directory` parameter set. For example, the following shows how to export a schema named `SALES` in an Autonomous Database named `DB2022ADB` with 16 OCPUs:

```
expdp sales/password@db2022adb_high
directory=data_pump_dir
dumpfile=exp%U.dmp
parallel=16
encryption_pwd_prompt=yes
filesize=1G
logfile=export.log
```

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

After the export is finished you can see the generated dump files by running the following query:

```
SELECT * FROM DBMS_CLOUD.LIST_FILES('DATA_PUMP_DIR');
```

For example, the output from this query shows the generated dump files and the export log file:

<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>
</tbody>
</table>
2. Move the dump file set to your cloud object store. See Move Dump File Set from Autonomous Database to Your Cloud Object Store for details.

Notes:

- To perform a full export or to export objects that are owned by other users, you need the DATAPUMP_CLOUD_EXP role.
- The DATA_PUMP_DIR is the only predefined directory. You can specify a different directory as the directory argument if you previously created the directory and you have write privileges on the directory. See Create Directory in Autonomous Database for information on creating directories.
- The API you use to move the dump files to Cloud Object Storage has a maximum file transfer size, so make sure you use a filesize argument that is less than or equal to the maximum supported size for your Cloud Object Storage service. See PUT_OBJECT Procedure for the Cloud Object Storage Service file transfer size limits.
- For more information on Oracle Data Pump Export see Oracle Database Utilities.

Move Dump File Set from Autonomous Database to Your Cloud Object Store

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

1. Connect to your database.
2. Store your object store credentials using the procedure DBMS_CLOUD.CREATE_CREDENTIAL.

For example:

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

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only
once unless your object store credentials change. Once you store the credentials you can then use the same credential name.

See CREATE_CREDENTIAL Procedure for information about the username and password parameters for different object storage services.

3. Move the dump files from the database to your Cloud Object Store by calling DBMS_CLOUD.PUT_OBJECT.

For example:

```sql
BEGIN
    DBMS_CLOUD.PUT_OBJECT(
        credential_name => 'DEF_CRED_NAME',
        object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp01.dmp',
        directory_name => 'DATA_PUMP_DIR',
        file_name => 'exp01.dmp');
    DBMS_CLOUD.PUT_OBJECT(
        credential_name => 'DEF_CRED_NAME',
        object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp02.dmp',
        directory_name => 'DATA_PUMP_DIR',
        file_name => 'exp02.dmp');
    DBMS_CLOUD.PUT_OBJECT(
        credential_name => 'DEF_CRED_NAME',
        object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp03.dmp',
        directory_name => 'DATA_PUMP_DIR',
        file_name => 'exp03.dmp');
    DBMS_CLOUD.PUT_OBJECT(
        credential_name => 'DEF_CRED_NAME',
        object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp04.dmp',
        directory_name => 'DATA_PUMP_DIR',
        file_name => 'exp04.dmp');
END;
/
```

In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

See PUT_OBJECT Procedure for information on PUT_OBJECT.

4. Perform the required steps to use Oracle Data Pump import and clean up. See Download Dump Files, Run Data Pump Import, and Clean Up Object Store for more details.

### Move Data with Data Pump Export to Object Store

To move data from Autonomous Databases to other Oracle databases you can use Oracle Data Pump. Using this export method you use Oracle Data Pump to directly export data to your object store. This export method is supported with Oracle Cloud Infrastructure Object Storage and Oracle Cloud Infrastructure Object Storage Classic.

See Move Data with Data Pump Export to an Autonomous Database Directory for details on the alternative export method to use with other supported cloud object stores.

To export data from Autonomous Databases to other Oracle databases, do the following:

1. Use Data Pump Export to export from your database to Object Storage.
You have two options for setting the credential to access Object Store, depending on your Oracle Data Pump version:

- Using Oracle Data Pump version 19.9 (or later) use the `CREDENTIAL` parameter to set the credential or set the database property `DEFAULT_CREDENTIAL`, both options are supported. See Use Oracle Data Pump to Export Data to Object Store Using CREDENTIAL Parameter (Version 19.9 or Later) for details.

- For Oracle Data Pump versions before 19.9 the `CREDENTIAL` parameter is not supported. Use the `DEFAULT_CREDENTIAL` database property to set the credential to use to access your Object Store. See Use Oracle Data Pump to Export Data to Object Store Setting DEFAULT_CREDENTIAL Property for details.

1. Connect to your 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',
   );
   END;
   ```

2. Download the dump files from the Object Storage service, run Data Pump Import with the dump files, and perform any required clean up such as removing the dump file set from Object Storage.

   See Download Dump Files, Run Data Pump Import, and Clean Up Object Store for details.

---

**Use Oracle Data Pump to Export Data to Object Store Using CREDENTIAL Parameter (Version 19.9 or Later)**

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

Oracle recommends using the latest Oracle Data Pump version for exporting data from Autonomous Database to other Oracle databases, as it contains enhancements and fixes for a better experience. Download the latest version of Oracle Instant Client and download the Tools Package, which includes Oracle Data Pump, for your platform from Oracle Instant Client Downloads. See the installation instructions on the platform install download page for the installation steps required after you download Oracle Instant Client and the Tools Package.

**Note:**

Database Actions provides a link for Oracle Instant Client. To access this link from Database Actions, under **Downloads**, click **Download Oracle Instant Client**.

If you are using Oracle Data Pump Version 19.9 or later, then you can use the `credential` parameter as shown in these steps. For instructions for using Oracle Data Pump Versions 19.8 and earlier, see Use Oracle Data Pump to Export Data to Object Store Setting DEFAULT_CREDENTIAL Property.

1. Connect to your database.

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

```sql
expdp admin/password@db2022adb_high \
    filesize=5GB \
    credential=def_cred_name \
    dumpfile=https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp%U.dmp \
    parallel=16 \
    encryption_pwd_prompt=yes \
    logfile=export.log \
    directory=data_pump_dir
```

**Note:**
If during the export with expdp you use the encryption_pwd_prompt=yes parameter then use encryption_pwd_prompt=yes with your import and input the same password at the impdp prompt to decrypt the dump files (remember the password you supply with export). The maximum length of the encryption password is 128 bytes.

In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

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 Database instance, as shown in the example.

For information on which database service name to connect to run Data Pump Export, see Database Service Names for Autonomous Data Warehouse.

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

This example shows the recommended parameters for exporting from Autonomous Database. For these expdp parameters, note the following:

- The maximum filesize parameter value is 10000MB for Oracle Cloud Infrastructure Object Storage exports.
• The maximum `filesize` parameter value is 20GB for Oracle Cloud Infrastructure Object Storage Classic exports.

• If the specified `filesize` is too large, the export shows the error message:

  ORA-17500: ODM err:ODM HTTP Request Entity Too Large

• The `directory` parameter specifies the directory `data_pump_dir` for the specified log file, `export.log`. See Access Log Files for Data Pump Export for more information.

---

**Note:**

Oracle Data Pump divides each dump file part into smaller chunks for faster uploads. The Oracle Cloud Infrastructure Object Storage console shows multiple files for each dump file part that you export. The size of the actual dump files will be displayed as zero (0) and its related file chunks as 10mb or less. For example:

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

Downloading the zero byte dump file from the Oracle Cloud Infrastructure console or using the Oracle Cloud Infrastructure CLI will not give you the full dump files. To download the full dump files from the Object Store, use a tool that supports Swift such as `curl`, and provide your user login and Swift auth token.

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

If you import a file with the `DBMS_CLOUD` procedures that support the `format` parameter type with the value 'datapump', you only need to provide the primary file name. The procedures that support the 'datapump' format type automatically discover and download the chunks.

When you use `DBMS_CLOUD.DELETE_OBJECT`, the procedure automatically discovers and deletes the chunks when the procedure deletes the primary file.

---

4. Perform the required steps to use Oracle Data Pump import and clean up.

See Download Dump Files, Run Data Pump Import, and Clean Up Object Store for details.
To perform a full export or to export objects that are owned by other users, you need the DATAPUMP_CLOUD_EXP role.

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

Use Oracle Data Pump to Export Data to Object Store Setting DEFAULT_CREDENTIAL Property

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

Oracle recommends using the latest Oracle Data Pump version for exporting data from Autonomous Database to other Oracle databases, as it contains enhancements and fixes for a better experience. Download the latest version of Oracle Instant Client and download the Tools Package, which includes Oracle Data Pump, for your platform from Oracle Instant Client Downloads. See the installation instructions on the platform install download page for the installation steps required after you download Oracle Instant Client and the Tools Package.

Note:
Database Actions provides a link for Oracle Instant Client. To access this link from Database Actions, under Downloads, click Download Oracle Instant Client.

1. Connect to your Autonomous Database.

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

```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. As the ADMIN user, set the credential you defined in step 2 as the default credential for your database. For example:

   ALTER DATABASE PROPERTY SET DEFAULT_CREDENTIAL = 'ADMIN.DEF_CRED_NAME'

4. Run Data Pump Export with the dumpfile parameter set to the URL for an existing bucket on your Cloud Object Storage (ending with a file name or a file name with a substitution variable, such as exp%U.dmp). For example:

   expdp admin/password@db2022adb_high \
   filesize=5GB \
   dumpfile=default_credential:https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp%U.dmp \
   parallel=16 \
   encryption_pwd_prompt=yes \
   logfile=export.log \n   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 Database, as shown in the example.

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

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

   This example shows the recommended parameters for exporting from Autonomous Database. For these expdp parameters, note the following:

   - The maximum filesize parameter value is 10000MB for Oracle Cloud Infrastructure Object Storage exports.
   - The maximum filesize parameter value is 20GB for Oracle Cloud Infrastructure Object Storage Classic exports.
   - If the specified filesize is too large, the export shows the error message:

     ORA-17500: ODM err:ODM HTTP Request Entity Too Large
• The directory parameter specifies the directory `data_pump_dir` for the specified log file, `export.log`. See Access Log Files for Data Pump Export for more information.

**Note:**

Oracle Data Pump divides each dump file part into smaller chunks for faster uploads. The Oracle Cloud Infrastructure Object Storage console shows multiple files for each dump file part that you export. The size of the actual dump files will be displayed as zero (0) and its related file chunks as 10mb or less. For example:

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

Downloading the zero byte dump file from the Oracle Cloud Infrastructure console or using the Oracle Cloud Infrastructure CLI will not give you the full dump files. To download the full dump files from the Object Store, use a tool that supports Swift such as curl, and provide your user login and Swift auth token.

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

If you import a file with the `DBMS_CLOUD` procedures that support the `format` parameter type with the value 'datapump', you only need to provide the primary file name. The procedures that support the 'datapump' format type automatically discover and download the chunks.

When you use `DBMS_CLOUD.DELETE_OBJECT`, the procedure automatically discovers and deletes the chunks when the procedure deletes the primary file.

5. Perform the required steps to use Oracle Data Pump import and clean up.
   See Download Dump Files, Run Data Pump Import, and Clean Up Object Store for details.

**Note:**

To perform a full export or to export objects that are owned by other users, you need the `DATAPUMP_CLOUD_EXP` role.

For detailed information on Oracle Data Pump Export parameters see Oracle Database Utilities.
Move Data to Object Store as Oracle Data Pump Files Using EXPORT_DATA

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

With this export method you use the DBMS_CLOUD.EXPORT_DATA procedure to specify a query to select the data to export, as follows:

1. Connect to your database.
2. Store your object store credentials using the procedure DBMS_CLOUD.CREATE_CREDENTIAL.

For example:

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

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name.

See CREATE_CREDENTIAL Procedure for information about the username and password parameters for different object storage services.

3. Export data from Autonomous Database to your Cloud Object Store as Oracle Data Pump dump file(s) by calling DBMS_CLOUD.EXPORT_DATA with the format parameter type set to value datapump. For example:

```
BEGIN
    DBMS_CLOUD.EXPORT_DATA(
        credential_name =>'DEF_CRED_NAME',
        file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp01.dmp',
        format => json_object(''type' value 'datapump'),
        query => 'SELECT warehouse_id, quantity FROM inventories'
    );
END;
/
```

The parameters are:
• credential_name: is the name of the credential created in the previous step.

• file_uri_list: is a comma delimited list of the export file(s). Use of wildcard and substitution characters is not supported in the file_uri_list.

• format: specifies the required type parameter with the value datapump, and optionally defines the options you can specify for the export with the ORACLE_DATAPUMP Access Driver.

• query: specifies a SELECT statement so that only the required data is exported. The query determines the contents of the dump file(s).

In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

Note:
The DBMS_CLOUD.EXPORT_DATA procedure creates the dump file(s) that you specify in the file_uri_list. The procedure does not overwrite files. If a dump file in the file_uri_list exists, DBMS_CLOUD.EXPORT_DATA reports an error. DBMS_CLOUD.EXPORT_DATA does not create buckets.

For detailed information about the parameters, see EXPORT_DATA Procedure.

4. Perform the required steps to use Oracle Data Pump import and clean up. See Download Dump Files, Run Data Pump Import, and Clean Up Object Store for more details.

Notes for exporting data with DBMS_CLOUD.EXPORT_DATA:

• The dump files you create with DBMS_CLOUD.EXPORT_DATA cannot be imported using Oracle Data Pump impdp. Depending on the database, you can use these files as follows:
  – On an Autonomous Database 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, to an Autonomous Transaction Processing database, or to Autonomous JSON Database.

If you export directly to Object Store using Oracle Data Pump, as shown in Move Data with Data Pump Export to Object Store, then the dump files on Object Store show size 0. Oracle Data Pump divides each dump file part into smaller chunks for faster uploads. The Oracle Cloud Infrastructure Object Storage console shows multiple files for each dump file part that you export. The size of the actual dump files will be displayed as zero (0) and its related file chunks as 10mb or less. For example:

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

Downloading the zero byte dump file from the Oracle Cloud Infrastructure console or using the Oracle Cloud Infrastructure CLI will not give you the full dump files. To download the full dump files from the Object Store, use a tool that supports Swift such as curl, and provide your user login and Swift auth token.

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

The cURL command does not support wildcards or substitution characters in its URL. You need to use multiple cURL commands to download the dump file set from your Object Store. Alternatively, you can use a script that supports substitution characters to download all the dump files from your Object Store in a single command. See How To: Download all files from an export to object store job in Autonomous Database using cURL for an example.

2. Run Data Pump Import to import the dump file set to the target database.
   If you are importing the data to another Autonomous Database, see Import Data Using Oracle Data Pump on Autonomous Database.
3. Perform post import clean up tasks. If you are done importing the dump files to your target database then drop the bucket containing the data or remove the dump files from the Cloud Object Store bucket, and remove the dump files from the location where you downloaded the dump files to run Data Pump Import.

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

Access Log Files for Data Pump Export

The log files for Data Pump Export operations are stored in the directory you specify with the data pump directory parameter.

To access the log file you need to move the log file to your Cloud Object Storage using the procedure DBMS_CLOUD.PUT_OBJECT. For example, the following PL/SQL block moves the file export.log to your Cloud Object Storage:

```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 => '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.

Move Data to Object Store as CSV, JSON, or XML Using EXPORT_DATA

Use DBMS_CLOUD.EXPORT_DATA to export data as text from an Autonomous Database. The text format export options are CSV, JSON, or XML.

Topics

- Export JSON Data to Cloud Object Storage
- Export Data as CSV to Cloud Object Storage
- Export Data as XML to Cloud Object Storage
- File Naming with Text Output (CSV, JSON, or XML)
Export JSON Data to Cloud Object Storage

Shows the steps to export table data from your Autonomous Database to Cloud Object Storage as JSON data by specifying a query.

This export method supports all the Cloud Object Stores supported by Autonomous Database, and you can use an Oracle Cloud Infrastructure resource principal to access your Oracle Cloud Infrastructure Object Store or Amazon Resource Names (ARNs) to access AWS Simple Storage Service (S3).

1. Connect to your Autonomous Database instance.
   See Connecting to Autonomous Database for more information.

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.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Run DBMS_CLOUD.EXPORT_DATA and specify the format parameter type with the value json to export the results as JSON files on Cloud Object Storage.

   To generate the JSON output files there are two options for the file_url_list parameter:
   • Set the file_url_list value to the URL for an existing bucket on your Cloud Object Storage.
   • Set the file_url_list value to the URL for an existing bucket on your Cloud Object Storage and include a file name prefix to use when generating the file names for the exported JSON.

   If you do not include the file name prefix in the file_url_list, DBMS_CLOUD.EXPORT_DATA supplies a file name prefix. See File Naming with Text Output (CSV, JSON, or XML) for details.

   For example, the following shows DBMS_CLOUD.EXPORT_DATA with a file name prefix specified in file_url_list:

   ```sql
   BEGIN
       DBMS_CLOUD.EXPORT_DATA(
           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.

For detailed information about the parameters, see EXPORT_DATA Procedure.

For detailed information about the available `format` parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see DBMS_CLOUD Package Format Options for EXPORT_DATA with Text Files (CSV, JSON, and XML).

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The `query` parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the `format` parameter with the `compression` option to compress the output files.
- When you no longer need the files that you export, use the procedure `DBMS_CLOUD.DELETE_OBJECT` or use native Cloud Object Storage commands to delete the files.

Export Data as CSV to Cloud Object Storage

Shows the steps to export table data from your Autonomous Database to Cloud Object Storage as CSV data by specifying a query.

This export method supports all the Cloud Object Stores supported by Autonomous Database, and you can use an Oracle Cloud Infrastructure resource principal to access your Oracle Cloud Infrastructure Object Store or Amazon Resource Names (ARNs) to access AWS Simple Storage Service (S3).

1. Connect to your Autonomous Database instance.
   
   See Connecting to Autonomous Database for more information.

2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'user1@example.com',
    password => 'password'
  );
END;
/
```

The values you provide for `username` and `password` depend on the Cloud Object Storage service you are using.
Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the `format` parameter type with the value `csv` to export the results as CSV files on Cloud Object Storage.

To generate the CSV output files there are two options for the `file_url_list` parameter:

- Set the `file_url_list` value to the URL for an existing bucket on your Cloud Object Storage.
- Set the `file_url_list` value to the URL for an existing bucket on your Cloud Object Storage and include a file name prefix to use when generating the file names for the exported CSV files.

If you do not include the file name prefix in the `file_url_list`, `DBMS_CLOUD.EXPORT_DATA` supplies a file name prefix. See File Naming with Text Output (CSV, JSON, or XML) for details.

For example, the following shows `DBMS_CLOUD.EXPORT_DATA` with a file name prefix specified in `file_uri_list`:

```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/dept_export',
        query           => 'SELECT * FROM DEPT',
        format          => JSON_OBJECT('type' value 'csv', 'delimiter' value '|', 'compression' value 'gzip'));
END;
/```

In this example, `namespace-string` is the Oracle Cloud Infrastructure object storage namespace and `bucketname` is the bucket name. See Understanding Object Storage Namespaces for more information.

For detailed information about the parameters, see EXPORT_DATA Procedure.

For detailed information about the available `format` parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see DBMS_CLOUD Package Format Options for EXPORT_DATA with Text Files (CSV, JSON, and XML).

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The `query` parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the `format` parameter with the `compression` option to compress the output files.
- When you no longer need the files that you export, use the procedure `DBMS_CLOUD.DELETE_OBJECT` or use native Cloud Object Storage commands to delete the files.
Export Data as XML to Cloud Object Storage

Shows the steps to export table data from your Autonomous Database to Cloud Object Storage as XML data by specifying a query.

This export method supports all the Cloud Object Stores supported by Autonomous Database, and you can use an Oracle Cloud Infrastructure resource principal to access your Oracle Cloud Infrastructure Object Store or Amazon Resource Names (ARNs) to access AWS Simple Storage Service (S3).

1. Connect to your Autonomous Database instance.
   See Connecting to Autonomous Database for more information.

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.

   Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the `format` parameter type with the value `xml` to export the results as XML files on Cloud Object Storage.

   To generate the XML output files there are two options for the `file_url_list` parameter:

   • Set the `file_url_list` value to the URL for an existing bucket on your Cloud Object Storage.

   • Set the `file_url_list` value to the URL for an existing bucket on your Cloud Object Storage and include a file name prefix to use when generating the file names for the exported JSON.

   If you do not include the file name prefix in the `file_url_list`, `DBMS_CLOUD.EXPORT_DATA` supplies a file name prefix. See File Naming with Text Output (CSV, JSON, or XML) for details.

   For example, the following shows `DBMS_CLOUD.EXPORT_DATA` with a file name prefix specified in `file_uri_list`:

   ```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/dept_export',
   );
   END;
   ```
In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

For detailed information about the parameters, see EXPORT_DATA Procedure.

For detailed information about the available format parameters you can use with DBMS_CLOUD.EXPORT_DATA, see DBMS_CLOUD Package Format Options for EXPORT_DATA with Text Files (CSV, JSON, and XML).

Notes for exporting with DBMS_CLOUD.EXPORT_DATA:

- The query parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the format parameter with the compression option to compress the output files.
- When you no longer need the files that you export, use the procedure DBMS_CLOUD.DELETE_OBJECT or use native Cloud Object Storage commands to delete the files.

File Naming with Text Output (CSV, JSON, or XML)

Describes the file naming and output files for DBMS_CLOUD.EXPORT_DATA results with CSV, JSON, or XML output.

DBMS_CLOUD.EXPORT_DATA performs the query specified with the query parameter and sends the results to text files on Object Store (either CSV, JSON, or XML depending on the format type parameter). To speed up the procedure and to generate the output as fast as possible, DBMS_CLOUD.EXPORT_DATA divides its work. This means that, depending on system resources, when you run DBMS_CLOUD.EXPORT_DATA the procedure creates multiple output files in the Cloud Object Store bucket.

The format for each generated file is:

[FileNamePrefix | client_info_module_action]_sequenceNum_timestamp.format_extension.
[compression_extension]

- FileNamePrefix: (optional) If a FileNamePrefix is supplied, DBMS_CLOUD.EXPORT_DATA uses the file name prefix to generate file names for the results. The FileNamePrefix is specified using the text supplied after the bucket name in the file_uri_list parameter value.
  
You cannot provide multiple values for the FileNamePrefix in the file_uri_list.

- client_info_module_action: If a file name prefix is not supplied with the file_uri_list parameter, DBMS_CLOUD.EXPORT_DATA uses the combination of client_info, application module and action as the file name prefix (when this information is available). The procedure obtains these names from the application information for the database session that runs the query. See
DBMS_APPLICATION_INFO for information on client_info, module name, and action name.

If a file name prefix is not supplied with the file_uri_list and the database session attributes are not available, DBMS_CLOUD.EXPORT_DATA uses the file name prefix "data".

- **sequenceNum**: The sequence number associated with the DBMS_CLOUD.EXPORT_DATA query. Depending on the query, the database service, and the number of OCPUs in the Autonomous Database instance, there are one or more sequenceNums. Also, depending on the size of the results, there are one or more output files for each sequenceNum.

See Manage Concurrency and Priorities on Autonomous Database for information on database services.

- **timestamp**: Timestamp when the file is uploaded.

- **format_extension**: The default value depends on the format type value:
  - CSV format: .csv
  - JSON format: .json
  - XML format: .xml

For more information, see the description for format option fileextension in DBMS_CLOUD Package Format Options for EXPORT_DATA with Text Files (CSV, JSON, and XML).

- **compression_extension**: When you include the format parameter with the compression option with the value gzip, this is "gz".

For example, the file name prefix in the following DBMS_CLOUD.EXPORT_DATA procedure is specified in the file_url_list parameter, as dept_export.

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/dept_export',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'json'));
END;
/
```

When you specify a file name prefix the generated output files include the file name prefix, similar to the following:

```
department_1_20210809T173033Z.json
department_2_20210809T173034Z.json
department_3_20210809T173041Z.json
department_4_20210809T173045Z.json
```

The number of generated output files depends on the size of the results, the database service, and the number of OCPUs in the Autonomous Database instance.
In the following example the file_uri_list parameter does not include a file name prefix and the compression parameter is supplied, with value gzip:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/',
    query           => 'SELECT * FROM DEPT',
    format          => json_object('type' value 'json', 'compression' value 'gzip'));
END;
/
```

When a file name prefix is not in the file_uri_list parameter, DBMS_CLOUD.EXPORT_DATA uses a file name prefix of the form: client_info_module_action. For this example the generated output files include the file name prefix that DBMS_CLOUD.EXPORT_DATA supplies and the files are compressed with gzip and the file extension .gz is added, as follows:

- Client1_Module1_Action1_1_20210809T173033Z.json.gz
- Client1_Module1_Action1_2_20210809T173034Z.json.gz
- Client1_Module1_Action1_3_20210809T173041Z.json.gz
- Client1_Module1_Action1_4_20210809T173035Z.json.gz

If the client_info_module_action session information is not available when you run DBMS_CLOUD.EXPORT_DATA, the file name prefix is set to data. For example:

- data_1_20210809T173033Z.json.gz
- data_2_20210809T173034Z.json.gz
- data_3_20210809T173041Z.json.gz
- data_4_20210809T173035Z.json.gz

Notes for file naming with DBMS_CLOUD.EXPORT_DATA:

- DBMS_CLOUD.EXPORT_DATA does not create buckets.
- The number of files that DBMS_CLOUD.EXPORT_DATA generates is determined by the number of OCPUs, the database service, and the size of the result data.
- When a generated file contains 10MB of data, a new output file is created. However, if you have less than 10MB of result data you may have multiple output files, depending on the database service and the number of OCPUs for the Autonomous Database instance.
- The default output file chunk size is 10MB. You can change this value with the format parameter maxfilesize option. See DBMS_CLOUD Package Format Options for EXPORT_DATA with Text Files (CSV, JSON, and XML) for more information.
Oracle GoldenGate Capture for Oracle Autonomous Database

Using Oracle GoldenGate you can capture changes from an Oracle Autonomous Database and replicate to any target database or platform that Oracle GoldenGate supports, including another Oracle Autonomous Database.

Oracle GoldenGate Capture for Oracle Autonomous Database supports the following:

- Replication for different use cases: Report Offloading, Active-Active, Cloud to Cloud, and Cloud to on-premise.
- Inter-region and cross-region replication: Replicate data between different Oracle Cloud data centers around the world
- Replicate between targets: Replicate from an Autonomous Database to any target database or platform that Oracle GoldenGate supports, including to other Oracle Autonomous Database environments.

See Using Oracle GoldenGate with Autonomous Database for more information.
Developing RESTful Services in Autonomous Database

You can develop and deploy RESTful Services with native Oracle REST Data Services (ORDS) support on Autonomous Databases. Simple Oracle Document Access (SODA) for REST lets you use a database as a simple JSON document store.

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
• Access Oracle REST Data Services, Oracle APEX, and Developer Tools Using a Vanity URL
• About Customer Managed Oracle REST Data Services on Autonomous Database

About Oracle REST Data Services in Autonomous Database

Oracle REST Data Services (ORDS) makes it easy to develop REST interfaces for relational data in a database. ORDS is a mid-tier Java application that maps HTTP(S) verbs, such as GET, POST, PUT, DELETE, and so on, to database transactions, and returns any results as JSON data.

Note:

The Oracle REST Data Services (ORDS) application in Autonomous Database is preconfigured and fully managed. ORDS connects to the database using the low predefined database service with a fixed maximum number of connections (the number of connections for ORDS does not change based on the number of OCPUs). It is not possible to change the default ORDS configuration. See About Customer Managed Oracle REST Data Services on Autonomous Database for information on using an additional alternative ORDS deployment that enables flexible configuration options.

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

See Predefined Database Service Names for Autonomous Database for information on the low database service.

Access RESTful Services and SODA for REST

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

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the Display Name column.

To use Oracle REST Data Services and SODA for REST:

1. From the Autonomous Database details page click Database Actions.
2. On the Database Actions launchpad, under Related Services, click the RESTful Services and SODA card to see the base URL.
3. Click Copy to copy the URL.

**Note:**

If you are using Always Free Autonomous Database with Oracle Database 21c, Oracle recommends the following:
For projects that were started using a database release prior to Oracle Database 21c, explicitly specify the metadata for the default collection as specified in the example in the section SODA Drivers. For projects started using release Oracle Database 21c or later, just use the default metadata. See SODA Drivers for more information.

Develop with Oracle REST Data Services on Autonomous Database

Autonomous Database supports Oracle REST Data Services (ORDS).

Developing RESTful services is easy with the following development interfaces:

- Database Actions (SQL Developer Web): Use Database Actions on Autonomous Database to REST enable users. See Manage Users and User Roles on Autonomous Database - Connecting with Database Actions for more information.
- SQL Developer (desktop): With SQL Developer on your desktop, you can connect to your database and enable REST services access to tables and views, or develop custom RESTful Services based on your SQL and PL/SQL code. See Connect Oracle SQL Developer with a Wallet (mTLS) for more information.
Oracle APEX (APEX): With APEX you can use the RESTful Services development pages to build and maintain your services and REST enabled objects. You can use the APEX SQL Workshop to access your Oracle RESTful Services and REST enabled objects. See How to Access RESTful Services for more information.

The Autonomous Database ADMIN account is REST Enabled. This allows for REST Services to be published in the ADMIN schemas and allows you to access Database Actions using the ADMIN database user account. Oracle recommends you create an application schema account for your RESTful Services and REST enabled objects. Services are secured using Database Authentication and your REST enabled schema.

The authenticated database user is only permitted access if the schema is REST enabled and the URL mapping for the request points to their own schema. A user is not authenticated when a request points to any other database schema. For example, the following request authenticated as the REST enabled schema HR is accessible:

```
GET /ords/hr/module/service/
```

However, when authenticated as the REST enabled schema SCOTT, the same request:

```
GET /ords/hr/module/service/
```

results in an error:

```
401 HTTP Unauthorized response/error
```

Any database user whose credentials are correct and meets these rules is authenticated and granted the ORDS, mid-tier, role: SQL Developer. The SQL Developer role enables the user to access any endpoint that requires the SQL Developer role.

See REST-Enable a Database Table in Quick Start Guide for information on how to enable a table for REST access.

Use SODA for REST with Autonomous Database

Autonomous Database supports Simple Oracle Document Access (SODA) for REST.

Topics

- Overview of Using SODA for REST
- Load Purchase-Order Sample Data Using SODA for REST
- Use SODA for REST with OAuth Client Credentials

Overview of Using SODA for REST

SODA for REST is a predeployed REST service that can be used to store JSON documents in a database. SODA enables flexible, NoSQL-style application development without having to use SQL. With SODA, JSON documents are stored in named collections and managed using simple CRUD operations (create, read, update and delete). And while SQL isn’t required, JSON stored in SODA collections is still fully accessible from SQL when needed. For example, an operational application may be fully built using SODA (without SQL) but then
the data may be later analyzed using SQL from outside of the application. Autonomous Database SODA gives application developers the best of the NoSQL and SQL worlds - fast, flexible, and scalable application development without losing the ability to leverage SQL for analytics and reporting.

SODA for REST is deployed in ORDS under the following URL pattern, where schema corresponds to a REST-enabled database schema.

/ords/schema/soda/latest/*

The following examples use the cURL command line tool (http://curl.haxx.se/) to submit REST requests to the database. However, other 3rd party REST clients and libraries should work as well. The examples use database schema ADMIN, which is REST-enabled. You can SODA for REST with cURL commands from the Oracle Cloud Shell.

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

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

These commands insert three JSON documents into the fruit collection:

> curl -X POST -u 'ADMIN:<password>' -H "Content-Type: application/json" --data '{"name":"orange", "count":42}' "https://example-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/fruit"

{"items":["id":"6F7E5C60197E4C8A83AC7D7654F2E375"...}

> curl -X POST -u 'ADMIN:<password>' -H "Content-Type: application/json" --data '{"name":"pear", "count":5}' "https://example-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/fruit"

{"items":["id":"83714B1E2BBA41F7BA4FA93B109E185"...}

> curl -X POST -u 'ADMIN:<password>' -H "Content-Type: application/json" --data '{"name":"apple", "count":12, "color":"red"}' "https://example-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/fruit"

{"items":["id":"BAD7EFA9A2AB49359B8F5251F0B28549"...}

This example retrieves a stored JSON document from the collection:

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

{...}
This SQL query accesses the fruit collection:

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

The query returns these three rows:

<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>

**Note:**

If you are using Always Free Autonomous Database with Oracle Database 21c, Oracle recommends the following:

For projects that were started using a database release prior to Oracle Database 21c, explicitly specify the metadata for the default collection as specified in the example in the section SODA Drivers. For projects started using release Oracle Database 21c or later, just use the default metadata. See SODA Drivers for more information.

These examples show a subset of the SODA and SQL/JSON features. See the following for more information:

- SODA for REST for complete information on Simple Oracle Document Access (SODA)
- SODA for REST HTTP Operations for information on the SODA for REST HTTP operations

Load Purchase-Order Sample Data Using SODA for REST
Oracle provides a substantial set of JSON purchase-order documents, in plain-text file POList.json, as a JSON array of objects, where each such object represents a document.

The following examples use the cURL command line tool (http://curl.haxx.se/) to submit REST requests to the database. However, other 3rd party REST clients and libraries should work as well. The examples use database schema ADMIN, which is REST-enabled. You can use SODA for REST with cURL commands from the Oracle Cloud Shell.

You can load this sample purchase-order data set into a collection purchaseorder on your Autonomous Database with SODA for REST, using these curl commands:

```bash
curl -X GET "https://raw.githubusercontent.com/oracle/db-sample-schemas/master/order_entry/POList.json" -o POList.json

curl -X PUT -u 'ADMIN:password' "https://example-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/purchaseorder"

curl -X POST -H -u 'ADMIN:password' 'Content-type: application/json' -d @POList.json "https://example-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/purchaseorder?action=insert"
```

You can then use this purchase-order data to try out examples in Oracle Database JSON Developer’s Guide.

For example, the following query selects both the id of a JSON document and values from the JSON purchase-order collection stored in column json_document of table purchaseorder. The values selected are from fields PONumber, Reference, and Requestor of JSON column json_document, which are projected from the document as virtual columns (see SQL NESTED Clause Instead of JSON_TABLE for more information).

```sql
SELECT id, t.*
FROM purchaseorder
NESTED json_document COLUMNS(PONumber, Reference, Requestor) t;
```

See the following for more information:

- SODA for REST for complete information on Simple Oracle Document Access (SODA)
- SODA for REST HTTP Operations for information on the SODA for REST HTTP operations

### Use SODA for REST with OAuth Client Credentials

You can access SODA for REST on Autonomous Database using OAuth authentication. Depending on your application, accessing SODA for REST with OAuth authentication can improve performance and security.

Perform the following steps to use OAuth authentication to provide limited access to SODA for REST on Autonomous Database:
1. As the ADMIN user, access Database Actions and create a user with the required privileges.
   a. Access Database Actions as ADMIN.
      See Access Database Actions as ADMIN for more information.
   b. In Database Actions, click ☰️ to show the available actions.
   c. In Database Actions, under Administration select Database Users.
   d. Click Create User.
   e. In the Create User area, on the User tab enter User Name and a Password and confirm the password.
   f. Select Web Access.
   g. In the Create User area, select the Granted Roles tab and grant DWROLE to the user.
   h. Click Create User.
      See Manage Users and User Roles on Autonomous Database - Connecting with Database Actions for more information.

2. Use a SQL worksheet in Database Actions to grant user privileges required to load data.
   a. Access Database Actions as ADMIN.
      See Access Database Actions as ADMIN for more information.
   b. In Database Actions, click ☰️ to show the available actions.
   c. In Database Actions, under Development click SQL to open a SQL worksheet.
   d. Grant user privileges required to load data to the user from Step 1.

      GRANT UNLIMITED TABLESPACE TO user_name;

      See Manage User Privileges on Autonomous Database - Connecting with a Client Tool for more information.

3. Sign out as the ADMIN user.

4. Sign in to Database Actions as the user that is setting up to use OAuth authentication.

5. In Database Actions, use a SQL worksheet to register the OAuth client.
   a. Register the OAuth client.
      For example, enter the following commands into the SQL worksheet, where you supply the appropriate values for your user and your client application.

      BEGIN
      OAUTH.create_client( 
        p_name => 'my_client',
        p_grant_type => 'client_credentials',
        p_owner => 'Example Company',
        p_description => 'A client for my SODA REST resources',
        p_support_email => 'user_name@example.com',
        p_privilege_names => 'my_priv'
      );

      OAUTH.grant_client_role(
        p_client_name => 'my_client',
      )
In the SQL worksheet, click Run Script to run the command.

See OAUTH PL/SQL Package Reference for more information.

This registers a client named my_client to access the my_priv privilege using OAuth client credentials.

6. Obtain the client_id and client_secret required to generate the access token.

For example, in the SQL worksheet run the following command:

```
SELECT id, name, client_id, client_secret FROM user_ords_clients;
```

7. Obtain the access token. To get an access token you send a REST GET request to database_ORDS_url/user_name/oauth/token.

The database_ORDS_url is available from Database Actions, under Related Services, on the RESTful Services and Soda card. See Access RESTful Services and SODA for REST for more information.

In the following command, use the client_id and the client_secret you obtained in Step 6.

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

You can use the cURL command line tool to submit the REST GET request. For example:

```
> curl -i -k --user SBA-i09Ke12cd2HYfryBQ..:vvUQ1AaqTqAgA2oN7afSg.. --data "grant_type=client_credentials"https://mqssyowmqvgac1y-doc.adb.region.oraclecloudapps.com/ords/user_name/oauth/token
HTTP/1.1 200 OK
Date: Mon, 22 Jun 2020 15:17:11 GMT
Content-Type: application/jsonTransfer-Encoding: chunked
Connection: keep-alive
X-Frame-Options: SAMEORIGIN

{"access_token":"JbOKtAuDgEh2DXx0QhvPGg","token_type":"bearer","expires_in":3600}
```

To specify both the client_id and the client_secret with the curl --user argument, enter a colon to separate the client_id and the client_secret. If you only specify the user name, client_id, curl prompts for a password and you can enter the client_secret at the prompt.
Use the access token to access the protected resource.

The token obtained in the previous step is passed in the Authorization header. For example:

```bash
> curl -i -H "Authorization: Bearer JbOKtAuDgEh2DXx0QhvPGg" -X GET https://database_id.odb.region.oraclecloudapps.com/ords/user_name/soda/latest
HTTP/1.1 200 OK
Date: Mon, 22 Jun 2020 15:20:58 GMT
Content-Type: application/json
Content-Length: 28
Connection: keep-alive
X-Frame-Options: SAMEORIGIN
Cache-Control: private,must-revalidate,max-age=0

{"items":[],"hasMore":false}
```

See Configuring Secure Access to RESTful Services for complete information on secure access to RESTful Services.

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

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

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

To learn more, see the following:

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

About Customer Managed Oracle REST Data Services on Autonomous Database

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. 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.

Installing and configuring a customer managed environment for ORDS is only supported with Autonomous Database on Shared Exadata Infrastructure. Your Autonomous Database must be configured for one of the following workload types: Data Warehouse, Transaction Processing, or JSON Database.

---

**Note:**

Oracle REST Data Services 19.4.6 or higher is required to use a customer managed environment for ORDS with Autonomous Database. Customer managed environment for ORDS is not supported if your database is configured for APEX workload type.
Using Oracle Database API for MongoDB

Oracle Database API for MongoDB makes it possible to connect to Oracle Autonomous Database using MongoDB language drivers and tools.

Oracle Database API for MongoDB leverages the converged database capabilities of an Autonomous Database to manage multiple data types, including JSON data, within a single database. For example, these converged database capabilities allow you to use SQL to query or update JSON data.

See Oracle Database API for MongoDB for more information.

See About Autonomous JSON Database for more information.

See About Autonomous Database for Transaction Processing and Mixed Workloads for more information.

Topics
- Configure Access for MongoDB
- User Management for MongoDB
- Create a Test Autonomous Database User for MongoDB
- Connect MongoDB Applications to Autonomous Database

Configure Access for MongoDB

Oracle Database API for MongoDB enables you to use an Oracle Autonomous Database as the data store. You can create and configure a new Autonomous Database or modify the configuration of an existing Autonomous Database.

Configure a New Autonomous Database for MongoDB

Follow the steps in Provision an Autonomous Database, up to the point where you select your Network Access Type.
At this point, to use Oracle Database API for MongoDB configure secure access with network access type Secure access from allowed IPs and VCNs only or Private endpoint access only.

Configure an Existing Autonomous Database for MongoDB

Open the Oracle Cloud Infrastructure Console for your Autonomous Database instance.

Note:

To use Oracle Database API for MongoDB the Network must be configured and the Access type must be either: Allow secure access from specified IPs and VCNs or Virtual Cloud Network.
Access Control List (ACL) Setup

See Configure Access Control Lists for an Existing Autonomous Database Instance for more information.

To configure your ACL for an IP address, you will need to obtain the public IP address. There are several ways to show your public IP address:

- In the choose network access area, click Add My IP Address. This copies your IP address into the Values field.
- After disabling any VPN, use the WhatIsMyIP website.
- After disabling any VPN, use the curl command: curl -s https://ifconfig.me.

Note:

Public IP addresses may change. Any change to your public IP address will require a change in the ACL. If you are unable to access your database, your ACL should be something you check.

ACLs Types and Use Cases

<table>
<thead>
<tr>
<th>ACL Type</th>
<th>Use Case</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Address</td>
<td>Local development laptops sharing the same public IP address</td>
<td>Easiest way to get started. Any laptop connected on this LAN will have access to the database with the database credentials.</td>
</tr>
<tr>
<td>CIDR Block</td>
<td>Local development laptop</td>
<td>Using IPv4/32 notation</td>
</tr>
<tr>
<td>IP Addresses separated by commas</td>
<td>Small number of local development laptops connected on distinct LANs (having distinct public IP addresses)</td>
<td>Can be tedious to manage with 10+ laptops.</td>
</tr>
<tr>
<td>CIDR Block</td>
<td>Local development laptops connected on the same subnet exposed to Internet (each laptop has its own public IP Address)</td>
<td>Rely on CIDR Block notation. See calculator here for more information. Example: 89.84.109.0/24 gives 256 possible IP addresses from 89.84.109.0 to 89.84.109.255</td>
</tr>
<tr>
<td>VCN with CIDR Block</td>
<td>For testing, production, or CI/CD pipeline hosted on OCI having their own VCN and Compute instances</td>
<td>Assign OCI compartment per environment type.</td>
</tr>
<tr>
<td>Mixing IP Address and VCN with CIDR Block</td>
<td>Local development laptop accessing a test Autonomous Database with connections from the testing environment or CI/CD pipeline</td>
<td>A common configuration option for on-going development work.</td>
</tr>
</tbody>
</table>
User Management for MongoDB

Oracle Database API for MongoDB enables you to use an Oracle Autonomous Database as the data store. If you want or need to use an existing Autonomous Database for this purpose, here is the workflow.

Oracle Database API for MongoDB enables the mapping of Autonomous Database objects to MongoDB objects as follows:

<table>
<thead>
<tr>
<th>MongoDB Object</th>
<th>Oracle Autonomous Database Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>schema</td>
</tr>
<tr>
<td>collection</td>
<td>table</td>
</tr>
<tr>
<td>document</td>
<td>document (in a column)</td>
</tr>
</tbody>
</table>

For example, you could create a collection using the Oracle Database API for MongoDB as follows:

```sql
use scott;
db.createCollection('fruit');
```

A table named **FRUIT** is created in the schema **SCOTT**.

When you connect to the Oracle Database API for MongoDB, you authenticate using an Autonomous Database username and password. This authenticated connection then accesses collections within the corresponding schema. This user must meet the following requirements:

- The user's schema must be ORDS-enabled, which is sometimes referred to as enabled for Web Access. See Basic Setup to Enable ORDS Database API for more information.
- The user must have the following roles and privileges: SODA_APP, CREATE TABLE, and CREATE SESSION. See Manage User Roles and Privileges on Autonomous Database for more information.
- The user has a quota on tablespace **DATA**. See Create Users on Autonomous Database for more information.

**Note:**

The role **DWROLE** in the Autonomous Database contains these roles, among others.

Access to schemas not granted to the user is prohibited. For example, the user **SCOTT** can only access collections in the schema **SCOTT**. There is one exception. If the authenticated user has the Autonomous Database privileges CREATE USER, ALTER USER and DROP USER, that user can access any ORDS-enabled schema.

Additionally, a user with these privileges can implicitly create schemas. That is, when the user creates a collection in a database that does not exist, the schema will be created automatically. See Oracle Database API for MongoDB for more information.
Create a Test Autonomous Database User for MongoDB

The steps that follow use Database Actions to create a test user with the proper roles. You can also use Database Actions SQL or other SQL command line utilities to execute the SQL statements directly. See Create Users on Autonomous Database - Connecting with a Client Tool for more information.

1. Open the Oracle Cloud Infrastructure Console for your Autonomous Database.
2. Select Database Actions.
3. From the Database Actions Launchpad, select Administration > Database Users.
5. Enter a User Name and Password for your test user. Select Web Access, and set the Quota on tablespace DATA.
6. Select the Granted Roles tab.
7. In addition to the default roles, select and add the SODA_APP role for the user.
8. Select the **Create User** button.

You can use this user, or a similarly authenticated user, for testing purposes. See [Test Connection Using the Command Line](#) for more information.

## Connect MongoDB Applications to Autonomous Database

Connecting your MongoDB application to Autonomous Database includes several steps, depending upon your requirements.

### Topics
- Retrieve the Autonomous Database Connection String
- Test Connection Using the Command Line
- Test Connection Using a Node.js Application

### Retrieve the Autonomous Database Connection String

Use the [MongoDB Shell](https://www.mongodb.com/) which is a command-line utility used to connect and query your data.
1. To retrieve the connection string for your Autonomous Database instance, open Database Actions. See Access Database Actions as ADMIN for more information.

2. On the Database Actions Launchpad, under Related Services, click Oracle Database API for MongoDB. For example:

3. On the Oracle Database API for MongoDB page click Copy. The Oracle Database API for MongoDB page shows two connection strings:
   - Copy the string with port 27017 if your driver supports the loadBalanced property.
   - Copy the string with port 27016 if your driver does not support the loadBalanced property.

Test Connection Using the Command Line

1. Login as the test user. See Create a Test Autonomous Database User for MongoDB for more information.

   $ mongosh --tls --tlsAllowInvalidCertificates 'mongodb://TESTUSER:<PASSWORD>@<database URL>.oci-region.oraclecloudapps.com:27017/admin?
authMechanism=PLAIN&authSource=$external&ssl=true&loadBalanced=false'
   
   Current Mongosh Log ID: 614c9e2a01e3575c8c0b2ec7
   
   Connecting to:        mongodb://TESTUSER:<PASSWORD>@<database 
URL>.oci-region.oraclecloudapps.com:27017/admin?
authMechanism=PLAIN&authSource=$external&ssl=true&loadBalanced=false
   
   Using MongoDB:       3.6.2
   Using Mongosh:       1.0.7
   
   For mongosh info see: https://docs.mongodb.com/mongodb-shell/admin
   > show dbs
   testuser              0 B
   >
**Note:**

Use URI percent-encoding to replace any reserved characters in your connection-string URI — in particular, characters in your username and password. These are the reserved characters and their percent encodings:

<table>
<thead>
<tr>
<th>!</th>
<th>#</th>
<th>$</th>
<th>%</th>
<th>&amp;</th>
<th>'</th>
<th>(</th>
<th>)</th>
<th>*</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td>%21</td>
<td>%23</td>
<td>%24</td>
<td>%25</td>
<td>%26</td>
<td>%27</td>
<td>%28</td>
<td>%29</td>
<td>%2A</td>
<td>%2B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>,</th>
<th>/</th>
<th>:</th>
<th>;</th>
<th>=</th>
<th>?</th>
<th>@</th>
<th>[</th>
<th>]</th>
</tr>
</thead>
<tbody>
<tr>
<td>%2C</td>
<td>%2F</td>
<td>%3A</td>
<td>%3B</td>
<td>%3D</td>
<td>%3F</td>
<td>%40</td>
<td>%5B</td>
<td>%5D</td>
</tr>
</tbody>
</table>

For example, if your username is RUTH and your password is @least1/2#: then your MongoDB connection string to server `<server>` might look like this:

'mongodb://RUTH:40least1%2F%23%3F@<server>:27017/ ruth/ ...'

Depending on the tools or drivers you use, you might be able to provide a username and password as separate parameters, instead of as part of a URI connection string. In that case you likely won’t need to encode any reserved characters they contain.

See also:

- Percent Encoding - Reserved Characters
- Uniform Resource Identifier (URI): Generic Syntax

2. Create a collection and insert documents into your collection.

```sql
> show collections

> db.createCollection( 'fruit' )
{ ok: 1 }
> show collections
fruit
> db.fruit.insertOne( {name:"orange", count:42} )
{ acknowledged: true,
  insertedId: ObjectId("614ca31fdab254f63e4c6b47") }
> db.fruit.insertOne( {name:"apple", count:12, color: "red"} )
{ acknowledged: true,
  insertedId: ObjectId("614ca340dab254f63e4c6b4b8") }
> db.fruit.insertOne( {name:"pear", count:5} )
```
Query the collection using a SQL client such as Database Actions.

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

Test Connection Using a Node.js Application

1. Download Node.js. If you have already downloaded or installed Node.js for your environment, you can skip this step.

   ```bash
   $ wget https://nodejs.org/dist/latest-v14.x/node-v14.17.5-linux-x64.tar.xz
   ```

2. Extract the contents of the Node.js archive. If you have already downloaded or installed Node.js for your environment, you can skip this step.

   ```bash
   $ tar -xf node-v14.17.5-linux-x64.tar.xz
   ```

3. Configure the `PATH` environment variable. If you have already downloaded or installed Node.js for your environment, you can skip this step.

   ```bash
   $ export PATH="`pwd`"/node-v14.17.5-linux-x64/bin:$PATH
   ```

4. Test your connection with a Javascript example.
a. Create a new directory.

$ mkdir autonomous_mongodb
$ cd autonomous_mongodb
$ npm init -y

b. Install mongodb dependency.

$ npm install mongodb

c. Create a JavaScript application named connect.js.

```javascript
const { MongoClient } = require("mongodb");
const uri = "mongodb://TESTUSER:<PASSWORD>@<Database URI>.<OCI region>.oraclecloudapps.com:27017/admin?
authMechanism=PLAIN&authSource=$external&ssl=true&loadBalanced=false";

const client = new MongoClient(uri);

async function run() {
  try {
    await client.connect();

    const database = client.db('admin');
    const movies = database.collection('movies');

    // Insert a movie
    const doc = { title: 'Back to the Future',
                 year: 1985, genres: ['Adventure',
                          'Comedy', 'Sci-Fi'] }
    const result = await movies.insertOne(doc);

    // Query for a movie that has the title 'Back to the Future'
    const query = { title: 'Back to the Future' };
    const movie = await movies.findOne(query);
    console.log(movie);
  } finally {
    // Ensures that the client will close when you finish/error
    await client.close();
  }
}
run().catch(console.dir);
```
**Note:**

Use URI percent-encoding to replace any reserved characters in your connection-string URI — in particular, characters in your username and password. These are the reserved characters and their percent encodings:

<table>
<thead>
<tr>
<th>!</th>
<th>#</th>
<th>$</th>
<th>%</th>
<th>&amp;</th>
<th>'</th>
<th>(</th>
<th>)</th>
<th>*</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td>%21</td>
<td>%23</td>
<td>%24</td>
<td>%25</td>
<td>%26</td>
<td>%27</td>
<td>%28</td>
<td>%29</td>
<td>%2A</td>
<td>%2B</td>
</tr>
<tr>
<td>,</td>
<td>/</td>
<td>:</td>
<td>;</td>
<td>=</td>
<td>?</td>
<td>@</td>
<td>[</td>
<td>]</td>
<td></td>
</tr>
<tr>
<td>%2C</td>
<td>%2F</td>
<td>%3A</td>
<td>%3B</td>
<td>%3C</td>
<td>%3D</td>
<td>%3E</td>
<td>%3F</td>
<td>%40</td>
<td>%5B</td>
</tr>
</tbody>
</table>

For example, if your username is **RUTH** and your password is `@least1/2#?` then your MongoDB connection string to server `<server>` might look like this:

'`mongodb://RUTH:%40least1%2F%23%3F@<server>:27017/ruth/ ...'`

Depending on the tools or drivers you use, you might be able to provide a username and password as separate parameters, instead of as part of a URI connection string. In that case you likely won't need to encode any reserved characters they contain.

See also:

- Percent Encoding - Reserved Characters
- Uniform Resource Identifier (URI): Generic Syntax

d. Run the example. The output should look similar to the following.

```
$ node connect
{
   _id: new ObjectId("611e3266005202371acf27c1"),
   title: 'Back to the Future',
   year: 1985,
   genres: [ 'Adventure', 'Comedy', 'Sci-Fi' ]
}
```
Creating Applications with Oracle APEX in Autonomous Database

You can create applications with Oracle APEX on Autonomous Database.

Topics

- About Oracle APEX
- Access Oracle APEX Administration Services
- Create Oracle APEX Workspaces in Autonomous Database
- Access Oracle APEX App Builder
- Create Oracle APEX Developer Accounts
- Load Data from the Cloud into Oracle APEX
- Use JSON Data with Oracle APEX
- Use Web Services with Oracle APEX
- Send Email from Oracle APEX
- Control Oracle APEX Upgrades
- Access Oracle APEX, Oracle REST Data Services, and Developer Tools Using a Vanity URL
- Restrictions and Limitations for Oracle APEX with Autonomous Database

About Oracle APEX

Oracle APEX (APEX) is a low-code development platform that enables you to build scalable, secure enterprise applications with world-class features that can be deployed anywhere.

Oracle APEX provides you with an easy-to-use browser-based environment to load data, manage database objects, develop REST interfaces, and build applications which look and run great on both desktop and mobile devices. You can use Oracle APEX to develop a wide variety of solutions: import spreadsheets and develop a single source of truth in minutes, create compelling data visualizations against your existing data, deploy productivity applications to elegantly solve a business need, or build your next mission-critical data management application.

Oracle APEX embraces SQL. Anything you can express with SQL can be easily employed in an Oracle APEX application. Oracle APEX also embodies low code with powerful data management and data visualization components, as well as responsive development out of the box. Instead of writing code by hand, you are able to use intelligent wizards to guide you through the rapid creation of applications and components.

Oracle APEX on Autonomous Database provides a preconfigured, fully managed and secured environment to both build and deploy world-class data-centric applications. There are no limits on the number of developers or end users for your Oracle APEX applications;
Autonomous Database can instantly scale compute and storage online as needed, based upon your workload. Additionally, Oracle APEX applications developed on-premise can be easily deployed to Oracle APEX on Autonomous Database, or vice-versa.

Configuration, patching, monitoring, and upgrading of all Oracle APEX components is fully managed by Oracle, leaving you free to focus on developing your solutions and solving your business problems. With Oracle APEX and low code, your organization can be more agile and develop solutions faster, for less cost, and with greater consistency. You can adapt to changing requirements with ease. And you can empower professional developers and everyone else in your organization to be a part of the solution.

This chapter covers information on Oracle APEX specific to working on Autonomous Database.

For more information on APEX, see the following:

- Oracle APEX Release 22.1
- apex.oracle.com

### Access Oracle APEX Administration Services

Each Autonomous Database instance includes a dedicated instance of Oracle APEX; you can use this instance to create multiple workspaces. A workspace is a shared work area where you can build applications. You create workspaces in Oracle APEX Administration Services.

> **Note:**

If your Autonomous Database is configured to use a Private Endpoint, then you can only access Oracle APEX from clients in the same Virtual Cloud Network (VCN). See [Configuring Network Access with Private Endpoints](#) for more information.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the `≡` next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the Display Name column.

To access Oracle APEX Administration Services you can use the Oracle Cloud Infrastructure Console or Database Actions.

**To access Administration Services from Database Actions:**

1. On the Autonomous Database details page click Database Actions.
2. From the Database Actions Launchpad, under **Development**, click **APEX**.

   The Oracle APEX Administration Services sign-in page appears.

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

3. In the **Password** field, enter the password for the ADMIN user.

   **Note:**

   Administration Services and the Oracle APEX development environment on Autonomous Database use Database Accounts authentication. This authentication method uses the database account user name and password to authenticate users.

   See [Set the ADMIN Password in Autonomous Database](#) to change the password.

4. Click **Sign In to Administration**.

   When you sign in for the first time, follow the prompts to create an Application Express workspace. See [Create Oracle APEX Workspaces in Autonomous Database](#) for more information.
To access Administration Services from Oracle Cloud Infrastructure Console:

1. On the Autonomous Database details page click the **Tools** tab.

2. In the Oracle APEX area, click **Open APEX**.

You can also use Administration Services to manage your APEX instance. See Oracle APEX Administration Services in *Oracle APEX Administration Guide* for more information.

### Create Oracle APEX Workspaces in Autonomous Database

An Autonomous Database instance does not have precreated Oracle APEX workspaces. Create a workspace if you have not already done so or use these instructions to create additional workspaces.

To create an Oracle APEX workspace:

1. Sign in to Oracle APEX Administration Services.

   See *Access Oracle APEX Administration Services* for more information.

2. Click **Create Workspace**.

3. On the Create Workspace page, in the **Database User** field, enter a new database username or choose an existing user from the list.

   The ADMIN database user cannot be associated with a workspace.
4. In the Password field, provide a strong password if the database user is a new user. If the user is an existing database user you do not enter a password.
   See About User Passwords on Autonomous Database to learn more about the default password complexity rules.

5. (optional) In the **Workspace Name** field, change the name of the workspace that was automatically populated.

6. Click **Create Workspace**.
   See Access Oracle APEX App Builder and Create Oracle APEX Developer Accounts to create additional developer accounts.

**Access Oracle APEX App Builder**

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

If your Autonomous Database is configured to use a Private Endpoint, you can only access Oracle APEX App Builder from clients in the same Virtual Cloud Network (VCN).


To access Oracle APEX App Builder:

1. Sign in to Oracle APEX using the workspace name, username, and password you specify when you create the workspace.

   ![Note](image)

   Oracle APEX Administration Services and the Oracle APEX development environment on Autonomous Database use Database Accounts authentication. This authentication method uses the database account user name and password to authenticate users.

2. On the Workspace home page, click the App Builder icon.
   See Create Oracle APEX Developer Accounts to create developer accounts.

**Create Oracle APEX Developer Accounts**

Oracle APEX developers need a developer account in each workspace where they wish to build applications. The initial developer account is created when you create a workspace (this account also has Workspace Administrator privilege). These steps show you how to create additional developer accounts for members of your team or reset their passwords. When you create a developer account, a corresponding database user is automatically created.

To create developer accounts and provide direct access to Oracle APEX:

1. Sign in to Oracle APEX Administration Services.
   See Access Oracle APEX Administration Services for more information.

2. Click **Manage Workspaces**.

3. Under Workspace Actions, click **Manage Developers and Users**.
4. On the Manage Application Developers and Users page, click **Create User**.

5. On the Create/Edit User page, in the **Username** field, enter a username.

6. In the **Email Address** field, enter an email address.

7. (Optional) Use the on-screen and in-line help to fill in additional fields.

8. In the **User is an administrator** field, select **No**.

9. In the **User is a developer** field, select **Yes**.

10. In the **Password** field, enter a strong password.
    
    See *Create Users on Autonomous Database - Connecting with a Client Tool* to learn more about the default password complexity rules.

11. In the **Confirm Password** field, confirm the password.

12. At the top of the page, click **Create User**.

    Alternatively, click **Create and Create Another** if you want to create the user and create another user.

To share sign-in details with developers:

1. On the Autonomous Databases page, under the **Display Name** column, select the Autonomous Database instance that contains the user and the user's workspace.

2. On the Autonomous Database Details page click **Database Actions**.

3. In Database Actions Launchpad, under **Development**, right-click **APEX** and choose **Copy URL**.

4. Provide the copied URL, along with the Workspace Name, the Username, and the Password for the developer account you created.

Using this URL developers can access the Oracle APEX environment without having to navigate to the Autonomous Database Oracle Cloud Infrastructure Console.

---

**Note:**

Changing the password of Workspace Administrators and Developers through **Manage Developers and Users** page or **Edit Profile** page only affects applications configured with "Application Express Accounts" authentication scheme. To change the password used to access App Builder, use Database Actions or another client to change the password of the corresponding database user.

See *Creating User Accounts in Oracle APEX Administration Guide* for more information.

---

**Load Data from the Cloud into Oracle APEX**

Using Oracle APEX SQL Workshop, you can declaratively load data from Object Store into the database schema associated with your Oracle APEX workspace.

To load data from the cloud, do the following:

1. Open your Oracle APEX workspace.
2. Select **SQL Workshop**.
3. Select **Utilities**.
4. Select **Data Workshop**.

   This shows the Load from Cloud button.

   ![Image of Get Started](image)

5. Under Get Started, select **Load from Cloud**.

   See Loading Data From the Cloud for more information.

---

### Use JSON Data with Oracle APEX

You can use Oracle APEX to create applications with JSON data. You must first create a view to extract the required attributes from the JSON data and maps them into columns of a relational view.

**Topics**

- Create a View from JSON Data Guide
- Create a View with JSON_TABLE Function

### Create a View from JSON Data Guide

Oracle APEX interprets data in relational format. Creating a view extracts required attributes from the JSON data and maps them into columns of a relational view.

For creating a view of JSON data that is stored in SODA collections, you can use SODA APIs and JSON Data Guide. The following PL/SQL code uses SODA APIs to create a Data Guide view on JSON Data stored in SODA Collections.

Run the following code in Oracle APEX SQL Workshop to create a view named `myview`:

```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 JSON_TABLE Function

You can create views of JSON data using the json_table SQL/JSON function.

The json_table SQL/JSON function projects specific JSON data to columns of various SQL data types. You can use the json_table function to map parts of a JSON document into the rows and columns of a new, virtual table, which you can also think of as an inline view.

See Create View on JSON Data for more information on creating views over JSON Data.

Use Web Services with Oracle APEX

You can interact with both SOAP and RESTful style web services from Oracle APEX (APEX) in your Autonomous Database instance.

Web services enable applications to interact with one another over the web in a platform-neutral, language independent environment. In a typical web services scenario, a business application sends a request to a service at a given URL by using the HTTP protocol. The service receives the request, processes it, and returns a response. Web services are typically based on Simple Object Access Protocol (SOAP) or Representational State Transfer (REST) architectures.

Using REST Data Sources (formerly called Web Source Modules), APEX developers can declaratively access data services from a variety of REST endpoints, allowing both read and write operations. In addition to supporting smart caching rules for remote REST data, Oracle APEX also offers the unique ability to directly manipulate the results of REST data sources using industry standard SQL.

The APEX_WEB_SERVICE package enables you to integrate other systems with APEX by allowing you to interact with web services anywhere you can use PL/SQL in your
application. The package contains procedures and functions to call both SOAP and RESTful style web services, and to simplify implementation of OAuth 2.0 flows.

Note the following when working with web services in APEX with Autonomous Database:

- All web services must be secured. Only HTTPS services are supported on the default port (443). Connections through IP addresses are not allowed. Your APEX instance is preconfigured with an Oracle Wallet that contains more than 90 of the most common trusted root and intermediate SSL certificates. The APEX_WEB_SERVICE package automatically takes advantage of this Oracle Wallet without additional configuration from application developers.

- Oracle APEX is able to access web services over the internet with no additional configuration required. In order to reach endpoints in private subnets or behind on-premises firewalls, ensure they meet the prerequisites from this section, Submit an HTTP Request to a Private Host with UTL_HTTP, and add the following access control list for the desired host:

```
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'www.example.com',
    ace => XS$ACE_TYPE(
      privilege_list => XS$NAME_LIST('http'),
      principal_name => APEX_APPLICATION.g_flow_schema_owner,
      principal_type => XS_ACL.ptype_db),
    private_target => true);
END;
/
```

**Note:** If you set ROUTE_OUTBOUND_CONNECTIONS database property to PRIVATE_ENDPOINT, you do not need to define access control lists for individual hosts in order to access them from APEX. Enhanced Security for Outbound Connections with Private Endpoints

- Each Autonomous Database instance is preconfigured with a network access control list (ACL) to permit outbound web service calls from Oracle APEX to public endpoints.

- Your APEX instance does not require an outbound web proxy. Attempting to use a web proxy at the API call level or application level shows the error: ORA-01031: insufficient privileges.

- There is a default limit of 50,000 outbound web service requests per APEX workspace in a rolling 24-hour period. If the limit of outbound web service calls is reached, the following SQL exception is raised on the subsequent request and the request is blocked:

```
ORA-20001: You have exceeded the maximum number of web service requests per workspace. Please contact your administrator.
```

You can raise or remove the default limit of outbound web service requests by setting a value for the MAX_WEBSERVICE_REQUESTS instance parameter or by updating the Maximum Web Service Requests attribute in APEX Administration Services. For example, to
change the limit to 250,000, connect to your database as ADMIN using a SQL client and execute the following:

```sql
BEGIN
    APEX_INSTANCE_ADMIN.SET_PARAMETER('MAX_WEBSERVICE_REQUESTS', '250000');
    COMMIT;
END;
/
```

To learn more, see:

- APEX_WEB_SERVICE in Oracle APEX API Reference
- Managing REST Data Sources in Oracle APEX App Builder User's Guide

### Send Email from Oracle APEX

You can use the APEX_MAIL package to send emails from Oracle APEX applications deployed in Autonomous Database.

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

![Note:]

Third-party email providers are not supported.

To enable APEX_MAIL functionality in your APEX instance in Autonomous Database:

1. Identify the SMTP connection endpoint for Email Delivery. You configure the endpoint as the SMTP Host in your APEX instance in Step 4. You may need to subscribe to additional Oracle Cloud Infrastructure regions if Email Delivery is not available in your current region. See Configure SMTP Connection for more information.

2. Generate SMTP credentials for Email Delivery. Your APEX instance uses credentials to authenticate with Email Delivery servers when you send email. See Generate SMTP Credentials for a User for more information.

3. Create an approved sender for Email Delivery. You need to complete this step for all email addresses you use as the "From" with APEX_MAIL.SEND calls, as the Application Email From Address in your apps, or in the SMTP_FROM instance parameter. See Managing Approved Senders for more information.

4. Connect to your Autonomous Database as ADMIN user using a SQL client and configure the following SMTP parameters using APEX_INSTANCE_ADMIN.SET_PARAMETER:
   - `SMTP_HOST_ADDRESS`: Specifies the SMTP connection endpoint from Step 1.
   - `SMTP_USERNAME`: Specifies the SMTP credential user name from Step 2.
   - `SMTP_PASSWORD`: Specifies the SMTP credential password from Step 2.
• Keep default values for `SMTP_HOST_PORT` parameter (587) and `SMTP_TLS_MODE` parameter (STARTTLS).

For example:

```sql
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 APEX instance:
   a. Sign in to APEX Administration Services.
   b. Open the Manage Instance page.
   c. Click the Mail Queue link in the Manage Meta Data section.

Alternatively, query `APEX_MAIL_QUEUE` and `APEX_MAIL_LOG` views using a SQL client.

---

**Note:**

There is a default limit of 5,000 emails per workspace in a 24-hour period. You can update or remove this limit in Oracle APEX Administration Services or by setting the `WORKSPACE_EMAIL_MAXIMUM` instance parameter. Oracle Cloud Infrastructure Email Delivery may impose additional limitations.
An approved sender must be set up for all “From:” addresses sending mail through Oracle Cloud Infrastructure, or mail will be rejected. There are limitations on approved senders with Oracle Cloud Infrastructure Email Delivery. See Managing Approved Senders for more information.

For more information, see:

- Overview of the Email Delivery Service
- APEX_MAIL in Oracle APEX API Reference
- APEX_INSTANCE_ADMIN in Oracle APEX API Reference

Control Oracle APEX Upgrades

By default, Autonomous Database applies Oracle APEX upgrades as soon as they are released in your region. You can set an option to defer upgrades for major Oracle APEX releases, such as an upgrade from 20.1 to 20.2, for up to 45 days.

Note:
The latest Oracle APEX Patch Set Bundles are automatically applied on Autonomous Database and cannot be deferred.

Defer Oracle APEX Upgrades

Deferring Oracle APEX upgrades allows you to do the following:

- Control when Oracle APEX is upgraded on your Autonomous Database. While there should be minimal disruption on a live system, you can still choose to upgrade at a time or on a day with less activity, for example on a weekend day.
- Validate your applications against the new version of Oracle APEX before you upgrade your production system.

To validate and test before you upgrade your production system, do the following:

- Clone your production or test environment. See Clone an Autonomous Database Instance for more information.
- Upgrade Oracle APEX on the clone as specified in Apply Oracle APEX Upgrades.
- Perform your testing and validation on the clone in a non-production environment.

To defer Oracle APEX upgrades, do the following:

1. Open Oracle APEX Administration Services.
   See Access Oracle APEX Administration Services for more information.
2. Open the Available Updates dialog by clicking the ⚙ next to Available Updates.
   This shows the Available Updates dialog.
3. In the Available Updates dialog select the Defer Upgrade option, Yes.
4. Click **Apply Changes**.

Notes for deferred Oracle APEX upgrades:

- If the Defer Upgrade setting is **Yes** and updates are available, then on the date indicated in the Available Updates dialog, Oracle APEX updates are automatically applied to your Autonomous Database instance. You cannot postpone the updates beyond the listed date.

- If the Defer Upgrade setting is **Yes** and updates are available, you cannot schedule the Oracle APEX upgrades for a specific date and time. When you are ready to upgrade, access Oracle APEX Administration Services and click **Upgrade Now**. See [Apply Oracle APEX Upgrades](#) for details.

- If your Autonomous Database is stopped at the time when the new Oracle APEX release becomes available in your region, the update is automatically applied the next time you start your database, regardless of the Defer Upgrade setting.

- When you change the Defer Upgrade setting, the change applies to future upgrades and does not impact an Oracle APEX upgrade that is already available to your Autonomous Database.

- If you configure Customer Managed Oracle REST Data Services (ORDS) in your Autonomous Database instance, APEX upgrades are always deferred, regardless of the Defer Upgrade setting. See [Apply Oracle APEX Upgrades](#) to apply an available APEX upgrade with Customer Managed Oracle REST Data Services (ORDS).

  See [About Customer Managed Oracle REST Data Services on Autonomous Database](#) for more information.

**Apply Oracle APEX Upgrades**

When you have deferred upgrades and an upgrade is available, you have the option to apply the upgrade at any time before the specified upgrade date.

When Oracle APEX upgrades are not available the Available Updates area shows **System is up-to-date**.

When the Defer Upgrade setting is **Yes**, if an Oracle APEX upgrade is available and you take no action, the upgrade is applied automatically on the date specified in the Available Updates area.
If you previously deferred upgrades as specified in Defer Oracle APEX Upgrades, then you can upgrade at any time, as follows:

1. Open Oracle APEX Administration Services.

2. Open the Available Updates dialog by clicking the next to Available Updates.

3. On the Available Updates dialog click Upgrade Now.

This starts the Oracle APEX upgrade process in the background. You can continue using Administration Services and App Builder as well as running apps during most of the upgrade process.
Access Oracle APEX, Oracle REST Data Services, and Developer Tools Using a Vanity URL

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

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

To learn more, see the following:

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

Restrictions and Limitations for Oracle APEX with Autonomous Database

This section lists the feature restrictions and limitations of Oracle APEX when used within the context of Autonomous Database. Certain limitations are required to protect the security and performance of your Oracle APEX environment.

- **Disabled options in Administration Services:**
  Some Oracle APEX Administration Services 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").
  - Manage Requests - Ability to enable, submit, and approve self-service workspace requests and change requests. For example, Manage Instance, Self-Service Sign Up.
  - Manage Instance, Instance Settings - Security, Instance Settings, and Workspace Purge Setting. Workspace Isolation:
  * Workspace Purge Settings.
  - 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
• Disabled options in Workspace Administration:
  – Make a Service Request - Ability to make service requests for schemas, storage, and termination.
• The following application authentication schemes are supported with limitations:
  – HTTP Header Variable: Only after configuring a vanity URL. See Access Oracle APEX, Oracle REST Data Services, and Developer Tools Using a Vanity URL to learn more.
  – LDAP Directory, including APEX_LDAP PL/SQL API: With the same restrictions that apply to the DBMS_LDAP PL/SQL package. See Restrictions and Notes for Database PL/SQL Packages for more information.
• Only the following APEX_INSTANCE_ADMIN procedures and functions are supported:

  ADD_SCHEMA  
  ADD_WORKSPACE  
  GET_PARAMETER  
  REMOVE_SCHEMA  
  REMOVE_WORKSPACE  
  SET_PARAMETER  
  VALIDATE_EMAIL_CONFIG

  See APEX_INSTANCE_ADMIN in Oracle APEX API Reference for more information.
• SMTP APEX instance parameters and certain others may be set using the APEX_INSTANCE_ADMIN package.
  See Available Parameter Values in Oracle APEX API Reference for more information.
• Oracle APEX is only available as a Full Development environment. Converting into a Runtime environment, which minimizes the installed software footprint and removes UI components such as App Builder and Administration Services, is not supported.
Creating and Managing Directories on Autonomous Database

Autonomous Database includes a predefined `data_pump_dir` directory in the database 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.
Notes:

- **CREATE DIRECTORY** creates the database directory object in the database and also creates the file system directory. For example the directory path could be:

  /u03/dbfs/7C149E35BB1000A45FD/data/stage

- You can create a directory in the root file system to see all the files with the following commands:

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

  After you create the **ROOT_DIR** directory, use the following command to list all files:

  ```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.

- Space in the file system allocated for the directories you create and their contents is part of your storage allocation. See **Database Monitor Overview** to view the total space allocated.

## Drop Directory in Autonomous Database

Use the database **DROP DIRECTORY** command to drop a directory object.

For example, the following command drops the database directory object **staging**:

```sql
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. Autonomous Database manages the underlying file system directory; users do not remove the file system directory.

List Contents of Directory in Autonomous Database

Use the function `DBMS_CLOUD.LIST_FILES` to list the contents of a directory.

For example, to list the contents of the `staging` directory, run the following query:

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

To run `DBMS_CLOUD.LIST_FILES` with a user other than ADMIN you need to grant read privileges on the directory to that user. See `LIST_FILES Function` for more information.

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',
    object_uri => 'https://objectstorage.usphoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/cwallet.sso',
```
directory_name => 'STAGE');
END;
/

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

To run DBMS_CLOUD.GET_OBJECT with a user other than ADMIN you need to grant write privileges on the directory to that user.

To run DBMS_CLOUD.PUT_OBJECT with a user other than ADMIN you need to grant read privileges on the directory to that user.

See GET_OBJECT Procedure and Function and PUT_OBJECT Procedure for more information.
Sending Email on Autonomous Database

There are a number of options for sending email on Autonomous Database, including the following:

- Use Email Delivery Service to send mail. See Send Email with Email Delivery Service on Autonomous Database for more information.
- Use an email provider that is on a Private Endpoint to send mail. See Send Email with an Email Provider on a Private Endpoint for more information.
- Use Oracle APEX to send mail. See Send Email from Oracle APEX for more information.

Send Email with Email Delivery Service on Autonomous Database

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

```
Note:

Oracle Cloud Infrastructure Email Delivery Service is the only supported email provider for public SMTP endpoints.
```

To send email with Oracle Cloud Infrastructure Email Delivery Service:

1. Identify your SMTP connection endpoint for Email Delivery. You may need to subscribe to additional Oracle Cloud Infrastructure regions if Email Delivery is not available in your current region.

   For example, select one of the following for the SMTP connection endpoint:
   - smtp.us-phoenix-1.oraclecloud.com
   - smtp.us-ashburn-1.oraclecloud.com
   - smtp.email.uk-london-1.oci.oraclecloud.com
   - smtp.email.eu-frankfurt-1.oci.oraclecloud.com

   See Configure SMTP Connection for more information.

2. Generate SMTP credentials for Email Delivery. `UTL_SMTP` uses credentials to authenticate with Email Delivery servers when you send email.

   See Generate SMTP Credentials for a User for more information.

3. Create an approved sender for Email Delivery. Complete this step for all email addresses you use as the "From" with `UTL_SMTP.MAIL`.

   See Managing Approved Senders for more information.

4. Allow SMTP access for ADMIN user by appending an Access Control Entry (ACE).
For example:

BEGIN
  -- Allow SMTP access for user ADMIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'www.us.example.com',
    lower_port => 587,
    upper_port => 587,
    ace => xs$ace_type(privilege_list => xs$name_list('SMTP'),
                       principal_name => 'ADMIN',
                       principal_type => xs_acl.ptype_db));
END;
/

5. Create a PL/SQL procedure to send email.
   For example, see the sample code shown in SMTP Send Email Sample Code.

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

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

See UTL_SMTP for information on UTL_SMTP.
See PL/SQL Packages for UTL_SMTP restrictions with Autonomous Database.

### SMTP Send Email Sample Code

Shows sample code for sending email 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.octl.username';
  passwd varchar2(50) := 'password';
  msg_from varchar2(50) := 'adam@example.com';
  mailhost VARCHAR2(50) := 'smtp.us-ashburn-1.oraclecloud.com';

BEGIN
  mail_conn := UTL_smtp.open_connection(mailhost, 587);
  utl_smtp.starttls(mail_conn);
  UTL_SMTP.AUTH(mail_conn, username, passwd, schemes => 'PLAIN');
  utl_smtp.mail(mail_conn, msg_from);
  utl_smtp.rcpt(mail_conn, msg_to);
  UTL_smtp.open_data(mail_conn);
```
UTL_SMTP.write_data(mail_conn, 'Date: ' || TO_CHAR(SYSDATE, 'DD-MON-YYYY HH24:MI:SS') || UTL_TCP.crlf);
UTL_SMTP.write_data(mail_conn, 'To: ' || msg_to || UTL_TCP.crlf);
UTL_SMTP.write_data(mail_conn, 'From: ' || msg_from || UTL_TCP.crlf);
UTL_SMTP.write_data(mail_conn, 'Subject: ' || msg_subject || UTL_TCP.crlf);
UTL_SMTP.write_data(mail_conn, 'Reply-To: ' || msg_to || UTL_TCP.crlf ||
UTL_TCP.crlf);
UTL_SMTP.write_data(mail_conn, msg_text || UTL_TCP.crlf || UTL_TCP.crlf);

UTL_smtp.close_data(mail_conn);
UTL_smtp.quit(mail_conn);

EXCEPTION
WHEN UTL_smtp.transient_error OR UTL_smtp.permanent_error THEN
  UTL_smtp.quit(mail_conn);
  dbms_output.put_line(sqlerrm);
WHEN OTHERS THEN
  UTL_smtp.quit(mail_conn);
  dbms_output.put_line(sqlerrm);
END;
/

Where:

• **mailhost**: specifies the SMTP Connection Endpoint from Step 1 in *Send Email with Email Delivery Service on Autonomous Database*.

• **username**: specifies the SMTP credential username from Step 2 in *Send Email with Email Delivery Service on Autonomous Database*.

• **passwd**: specifies the SMTP credential password from Step 2 in *Send Email with Email Delivery Service on Autonomous Database*.

• **msg_from**: specifies one of the approved senders from Step 3 in *Send Email with Email Delivery Service on Autonomous Database*.

## Send Email with an Email Provider on a Private Endpoint

Describes the steps to send email with an email provider that is on Private Endpoint.

To send email from Autonomous Database using an email provider on a private endpoint, the email provider must be accessible from the Oracle Cloud Infrastructure VCN (the Autonomous Database instance's private endpoint). For example, you can access an email provider when:

• Both the source Autonomous Database instance and the email provider are in the same Oracle Cloud Infrastructure VCN.

• The source Autonomous Database instance and the email provider are in different Oracle Cloud Infrastructure VCNs that are paired.

• The email provider is on an on-premises network that is connected to the source Autonomous Database instance's Oracle Cloud Infrastructure VCN using FastConnect or VPN.

As a prerequisite, to send email using an email provider, define the following ingress and egress rules:
• Define an egress rule in the source database's subnet security list or network security group such that the traffic to the target host is allowed on port 587 or port 25 (depending on which port you are using).

• Define an ingress rule in the target host's subnet security list or network security group such that the traffic from the source Autonomous Database instance's IP address to port 587 or port 25 is allowed (depending on which port you are using).

Note:

Making calls to a private email provider is only supported in commercial regions and US Government regions. This feature is enabled by default in all commercial regions.

This feature is enabled by default in US Government regions for newly provisioned databases.

For existing US Government databases on a private endpoint, if you want to call an email provider from an Autonomous Database to a target in a US Government region, please file a Service Request at Oracle Cloud Support and request to enable the private endpoint in government regions database linking feature.

US Government regions include the following:

• Oracle Cloud Infrastructure US Government Cloud with FedRAMP Authorization
• Oracle Cloud Infrastructure US Federal Cloud with DISA Impact Level 5 Authorization

To send email from an email provider on private endpoint:

1. Allow SMTP access for ADMIN user by appending an Access Control Entry (ACE).

For example:

```sql
-- Create an Access Control List for the host
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'www.example.com',
    lower_port => 587,
    upper_port => 587,
    ace => xs$ace_type(Privilege_list => xs$name_list('SMTP'),
                       principal_name => 'ADMIN',
                       principal_type => xs_acl.ptype_db),
    private_target => TRUE);
END;
/```
Note:

DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE only supports a single hostname for the host parameter. Using an IP address or SCAN IP is not supported.

If you set ROUTE_OUTBOUND_CONNECTIONS to PRIVATE_ENDPOINT, setting the private_target parameter to TRUE is not required in this API. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

2. Create a PL/SQL procedure to send email.

3. Send a test email using the PL/SQL procedure you created in step 2.
   For example:

   ```sql
   execute send_mail('taylor@example.com', 'Email from Oracle Autonomous Database', 'Sent using private email provider');
   ```
Oracle Extensions for IDEs

Oracle extensions let developers connect to, browse, and manage Autonomous Databases directly from common 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 plugin that enables Java developers to easily connect to Oracle Autonomous Database through their IDE. The plugin is free and is available for Linux, UNIX, Microsoft Windows, and Apple Mac OS.

You can use the plugin to perform cloud and database operations right from Eclipse, such as creating Autonomous Databases, stopping and starting, scaling up and down, and so on. You can also use the plugin to easily connect to the databases to browse the schema, access tables, execute SQL statements, and perform other development tasks.

Users with permissions to manage the databases can perform a number of actions, including those listed below. For detailed information about permissions, see Toolkit for Eclipse in the Oracle Cloud Infrastructure documentation. You can:

- Create Autonomous Databases
- Start, stop, terminate, clone, and restore Autonomous Databases
- Scale up and down
- Download the client credentials zip file (database wallet)
- Connect to Autonomous Databases
- Browse the schema
- Choose compartments and regions
- Change the administrator password, update the license type, and so on

Download the latest version of the plugin from GitHub (com.oracle.oci.eclipse-version.zip, where version is the latest version, for instance 1.2.0):

https://github.com/oracle/oci-toolkit-eclipse/releases

Then follow the installation instructions and details about how to get started in this step-by-step walkthrough:

New Eclipse Plugin for Accessing Autonomous Database (ATP/ADW)
Use Oracle Developer Tools for Visual Studio

Oracle Developer Tools for Visual Studio is a tightly integrated extension for Microsoft Visual Studio and Oracle Autonomous Database. The extension is free and supports Visual Studio 2019 and Visual Studio 2017 on Microsoft Windows.

You can use the extension to perform database management operations right from Visual Studio, such as creating Autonomous Databases, stopping and starting, scaling up and down, and so on. You can also use the extension to easily connect to the databases and perform development tasks, such as browsing your Oracle schema and launching integrated Oracle designers and wizards to create and alter schema objects.

Users with permissions to manage the databases can perform a number of actions, including the following:

- Sign up for Oracle Cloud
- Connect to a cloud account using a simple auto-generated config file and key file
- Create new or clone existing Always Free Autonomous Database, Autonomous Database Dedicated, and Autonomous Database 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
- Scale up/down Autonomous Database resources
- Restore from backup
- Update instance credentials, update the license type used
- Rotate wallets
- Convert Always Free Autonomous Database into paid databases

**Note:**

Promotion of Always Free to paid is currently supported only if the Always Free instance has database release 19c.

Download the extension from Visual Studio Marketplace:

- Oracle Developer Tools for Visual Studio 2019
- Oracle Developer Tools for Visual Studio 2017

You'll find lots of information about the extension on those Marketplace pages.

Then follow the installation instructions and details about how to get started in this step-by-step walkthrough:

**New Release: Visual Studio Integration with Oracle Autonomous Database**

For detailed information about how to use the extension, see the online documentation that's optionally installed with Oracle Developer Tools for Visual Studio. Press the F1 key to display the context-sensitive help for each dialog.
Use Oracle Developer Tools for VS Code

Oracle Developer Tools for VS Code is a tightly integrated extension for Microsoft Visual Studio Code (VS Code) and Oracle Autonomous Database. The extension is free and is available for Linux, Microsoft Windows, and Apple Mac OS.

You can use the extension to connect to Autonomous Databases right from Visual Studio Code and easily explore database schema, view table data, and edit and execute SQL and PL/SQL.

Download the extension from Visual Studio Marketplace:

Oracle Developer Tools for VS Code

Installation instructions and information about how to get started can be found in this quick start guide:

Getting Started Using Oracle Developer Tools for VS Code
Using JSON Documents with Autonomous Database

Autonomous Database has full support for data represented as JSON documents. In Autonomous Databases, JSON documents can coexist with relational data.

You can access and manage JSON documents as follows:

- Using Simple Oracle Document Access (SODA) APIs for NoSQL-style access to document collections.
- Using SQL and PL/SQL to access JSON documents in tables.

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
- Import SODA Collection Data Using Oracle Data Pump Version 19.6 or Later

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

Simple Oracle Document Access (SODA) is a set of NoSQL-style APIs that let you use collections of JSON documents in Autonomous Database, retrieve them, and query them, without needing to know Structured Query Language (SQL) or how the documents are stored in the database.

Autonomous Database supports storing and querying JSON documents natively. SODA document collections are backed by ordinary database tables and views; you can take advantage of database features for use with the content of SODA documents.

SODA drivers are available for several languages and frameworks including: Java, Node.js, Python, C (using Oracle Call Interface), and PL/SQL, and SODA for REST. SODA for REST maps SODA operations to Uniform Resource Locator (URL) patterns, so it can be used with most programming languages.

To get started with SODA, see the following:

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

Depending on the SODA API you want to work with, see the following:
Note:

If you are using Always Free Autonomous Database with Oracle Database 21c, then to avoid compatibility problems of SODA drivers, Oracle recommends the following:

- Use the driver versions that are needed for working with JSON type as specified in SODA Drivers. See SODA Drivers for more information.
- For projects that were started using a database release prior to Oracle Database 21c, explicitly specify the metadata for the default collection as specified in the example in SODA Drivers. For projects started using release Oracle Database 21c or later, just use the default metadata. See SODA Drivers for more information.

<table>
<thead>
<tr>
<th>SODA API</th>
<th>Download and Installation</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SODA for Java</td>
<td>Download SODA for Java</td>
<td>Using SODA for Java</td>
</tr>
<tr>
<td></td>
<td>SODA for Java Prerequisites</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Versions:</strong> SODA for Java, using the latest version is recommended. The minimum supported version is: 1.1.4.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Use SODA for Java in conjunction with ojdbc8.jar for 19.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(available at Oracle Database 19c (19.6) JDBC Driver &amp; UCP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Downloads, or on Maven Central).</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Autonomous Database does not support Metadata builder. To</td>
<td></td>
</tr>
<tr>
<td></td>
<td>customize collection metadata pass collection metadata strings directly to the createCollection method.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>See SODA Collection Metadata on Autonomous Database for more</td>
<td></td>
</tr>
<tr>
<td></td>
<td>information.</td>
<td></td>
</tr>
<tr>
<td>SODA for REST</td>
<td>Access RESTful Services and SODA for REST</td>
<td>Use SODA for REST with Autonomous Database</td>
</tr>
<tr>
<td>SODA for C</td>
<td>Oracle Instant Client Downloads</td>
<td>Using SODA for C</td>
</tr>
<tr>
<td></td>
<td>SODA for C Prerequisites</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Versions:</strong> For SODA for C, Oracle Client libraries must be 19.6 and above. You can obtain Oracle Instant Client from Oracle Instant Client Downloads.</td>
<td></td>
</tr>
<tr>
<td>SODA for PL/SQL</td>
<td>No need to download. This is included with Autonomous Database.</td>
<td>Using SODA for PL/SQL</td>
</tr>
<tr>
<td></td>
<td>SODA for PL/SQL Prerequisites</td>
<td></td>
</tr>
<tr>
<td>SODA API</td>
<td>Download and Installation</td>
<td>More Information</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>SODA for Node.js</td>
<td>Multiple downloads described in install instructions: Quick Start node-oracledb Installation</td>
<td>Node-oracledb SODA Requirements</td>
</tr>
<tr>
<td></td>
<td><strong>Versions:</strong> 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 <a href="https://oracle.com/instantclient">Oracle Instant Client Downloads</a>.</td>
<td></td>
</tr>
<tr>
<td>SODA for Python</td>
<td>Multiple downloads described in install instructions: cx_Oracle 7 Installation</td>
<td>Introduction to cx_Oracle</td>
</tr>
<tr>
<td></td>
<td><strong>Versions:</strong> SODA support was introduced in version 7.0. Using the latest version is recommended, the minimum recommended version is 7.1. Oracle Client libraries must be 19.6 and above. You can obtain Oracle Instant Client from <a href="https://oracle.com/instantclient">Oracle Instant Client Downloads</a>.</td>
<td></td>
</tr>
</tbody>
</table>

**SODA Notes**

When you use SODA with Autonomous Database the following restrictions apply:

- Automatic indexing is not supported for SQL and PL/SQL code that uses the SQL/JSON function `json_exists`. See SQL/JSON Condition `JSON_EXISTS` for more information.
- Automatic indexing is not supported for SODA query-by-example (QBE).

**Work with JSON Documents Using SQL and PL/SQL APIs on Autonomous Database**

Autonomous Database supports JavaScript Object Notation (JSON) data natively in the database.

When you use Autonomous Database to store JSON data you can take advantage of all the features available in your database. You can combine your JSON data with non-JSON data; then, using Autonomous Database features such as Oracle Machine Learning Notebooks, you can analyze your data and create reports. You can access JSON data stored in the database the same way you access other database data, including using Oracle Call Interface (OCI), Microsoft .NET Framework, and Java Database Connectivity (JDBC).

See [JSON in Oracle Database](https://docs.oracle.com-en/database/19.3.0/tna/JSON.htm) to get started.
Load JSON Documents with Autonomous Database

You can use the PL/SQL procedure `DBMS_CLOUD.COPY_COLLECTION` to load JSON documents into SODA collections or use `DBMS_CLOUD.COPY_DATA` to load JSON documents into an existing table in Autonomous Database.

Topics

- About Loading JSON Documents
- Load a JSON File of Line-Delimited Documents into a Collection
- Load an Array of JSON Documents into a Collection
- Monitor and Troubleshoot COPY_COLLECTION Loads
- Create Credentials and Copy JSON Data into an Existing Table

Import SODA Collection Data Using Oracle Data Pump Version 19.6 or Later

Shows the steps to import SODA collections into Autonomous Database with Oracle Data Pump.

You can export and import SODA collections using Oracle Data Pump Utilities starting with version 19.6. Oracle recommends using the latest Oracle Data Pump version for importing data from Data Pump files into your database.

Download the latest version of Oracle Instant Client, which includes Oracle Data Pump, for your platform from Oracle Instant Client Downloads. See the installation instructions on the platform install download page for the installation steps required after you download Oracle Instant Client.

In Oracle Data Pump, if your source files reside on Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure native URIs, Swift URIs, or pre-authenticated URIs. See `DBMS_CLOUD Package File URI Formats` for details on these file URI formats.

If you are using an Oracle Cloud Infrastructure pre-authenticated URI, you still need to supply a credential parameter. However, credentials for a pre-authenticated URL are ignored (and the supplied credentials do not need to be valid). See `DBMS_CLOUD Package File URI Formats` for information on Oracle Cloud Infrastructure pre-authenticated URIs.

This example shows how to create the SODA collection metadata and import a SODA collection with Data Pump.

1. On the source database, export the SODA collection using the Oracle Data Pump `expdp` command.
   
   See Export Your Existing Oracle Database to Import into Autonomous Database for more information.

2. Upload the dump file set from Step 1 to Cloud Object Storage.

3. Create a SODA collection with the required SODA collection metadata on your Autonomous Database.
For example, if you export a collection named `MyCollectionName` from the source database with the following metadata:

- The content column is a **BLOB** type.
- The version column uses the **SHA256** method.

Then on the Autonomous Database where you import the collection create a new collection:

- By default on Autonomous Database for a new collection the content column is set to **BLOB** with the **jsonFormat** specified as **OSON**.
- By default on Autonomous Database for a new collection the **versionColumn.method** is set to **UUID**.

See [SODA Default Collection Metadata on Autonomous Database](#) for details.

```sql
DECLARE
collection_create SODA_COLLECTION_T;
BEGIN
collection_create := DBMS_SODA.CREATE_COLLECTION('MyCollectionName');
END;
/
COMMIT;
```

You can use the PL/SQL function `DBMS_SODA.LIST_COLLECTION_NAMES` to discover existing collections. See [LIST_COLLECTION_NAMES Function](#) for more information.

You can view the metadata for the SODA collections by querying the view `USER_SODA_COLLECTIONS`. See [USER_SODA_COLLECTIONS](#) for more information.

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

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

The values you provide for `username` and `password` depend on the Cloud Object Storage service you are using. For Oracle Cloud Infrastructure Object Storage, the `username` is your Oracle Cloud Infrastructure user name and `password` is your Oracle Cloud Infrastructure auth token. For more information on the credentials for different Cloud Object Storage services, see [CREATE_CREDENTIAL Procedure](#).

5. Run Data Pump Import with the `dumpfile` parameter set to the list of file URLs on your Cloud Object Storage and the `credential` parameter set to the name of the credential you created in the previous step.
Note:
Import the collection data using the option `CONTENT=DATA_ONLY`.

Specify the collection you want to import using the `INCLUDE` parameter. This is useful if a data file set contains the entire schema and the SODA collection you need to import is included as part of the dump file set.

Use `REMAP_DATA` to change any of the columns during import. This example shows using `REMAP_DATA` to change the version column method from `SHA256` to `UUID`.

```bash
impdp admin/password@db2022adb_high \
  directory=data_pump_dir \
  credential=def_cred_name \
  dumpfile= https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/bucketname/o/export%u.dmp \
  encryption_pwd_prompt=yes \
  SCHEMA=my_schema \
  INCLUDE=TABLE:"= "MyCollectionName"" \
  CONTENT=DATA_ONLY \

REMAP_DATA=my_schema.""MyCollectionName"".VERSION:SYS.DBMS_SODA.TO_UUID
```

Note:
If during the export with `expdp` you used the `encryption_pwd_prompt=yes` parameter then use `encryption_pwd_prompt=yes` and input the same password at the `impdp` prompt that you specified during the export.

In this example, `namespace-string` is the Oracle Cloud Infrastructure object storage namespace and `bucketname` is the bucket name. See Understanding Object Storage Namespaces for more information.

In Oracle Data Pump version 19.6 and later, the `credential` argument authenticates Oracle Data Pump to the Cloud Object Storage service you are using for your source files. The `dumpfile` argument is a comma delimited list of URLs for your Data Pump files.

For the best import performance use the `HIGH` database service for your import connection and set the `PARALLEL` parameter to the number of OCPUs in your Autonomous Database.

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

For the dump file URL format for different Cloud Object Storage services, see DBMS_CLOUD Package File URI Formats.
For information on disallowed objects in Autonomous Database, see [SQL Commands](#).

In this import example, the specification for the `REMAP_DATA` parameter uses the function `DBMS_SODA.TO_UUID` to generate UUID values. By default, for on-premise databases, the version column of a SODA collection is computed using SHA-256 hash of the document's content. On Autonomous Database the version column uses UUID generated values, which are independent of the document's content.

In this example the `REMAP_DATA` parameter uses the `DBMS_SODA.TO_UUID` function to replace the source collection version type with UUID versioning. If in the export dump file set that you are importing the `versionColumn.method` is already set to UUID, then the `REMAP_DATA` for this field is not required.

For detailed information on Oracle Data Pump Import parameters see [Oracle Database Utilities](#).

The log files for Data Pump Import operations are stored in the directory you specify with the Data Pump Import `DIRECTORY` parameter. See [Access Log Files for Data Pump Import](#) for more information.
Using Oracle Graph with Autonomous Database

Oracle Graph with Autonomous Database enables you to create graphs from data in your Autonomous Database. With graphs you can analyze your data based on connections and relationships between data entities.

As an Analyst or a Developer you can use graph algorithms and graph pattern queries for ranking, clustering, and path analysis in a graph model of your data. You can use graph features to detect anomalous patterns, identify communities, and find new connections in your data. Then, you can use graphs in your applications, for example, for fraud detection in banking, improved traceability in smart manufacturing, building linked data applications, and more; all while gaining enterprise-grade security, ease of data ingestion, and support for a wide range of workloads.

Autonomous Database includes Graph Studio, which automates graph data management and simplifies modeling, analysis, and visualization across the graph analytics lifecycle. Autonomous Database includes all the graph capabilities from Oracle Database, except the following:

• Apache Jena is not supported on Autonomous Database. See RDF Semantic Graph Support for Apache Jena for information on Oracle Database support for Apache Jena.

About Oracle Graph Studio on Autonomous Database

Graph Studio allows you to work with the two popular graph models, property graphs and RDF graphs.

Graph Studio features include automated modeling to create graphs from database tables, an integrated notebook to run graph queries and analytics, and native graph and other visualizations. You can easily create and manage either of the graph models. You can validate the graphs by executing queries and exploring their properties.

See Graph Studio: Interactive, Self-Service User Interface and Access Graph Studio Using Oracle Cloud Infrastructure Console for more information on Graph Studio.

Notes for Graph Studio on Autonomous Database:

• Graph Studio is supported with Data Warehouse and Transaction Processing workloads.

Building Applications Using Graph APIs with Autonomous Database

If you are developing an application using graphs, you can also access Property Graph APIs and RDF Graph APIs with Autonomous Database without Graph Studio.

• Property Graph: Create and work with Property Graphs with Python and Java developer APIs and the Property Graph Query Language (PGQL). You can use developer APIs to create a graph, run graph queries, and execute graph algorithms.
You can use these APIs by deploying Oracle Graph Server and Client from OCI Marketplace.

See Oracle Database Graph for more information.

- **RDF Graph**: Using RDF Graph Server and Query UI you can manage, store, query, and perform inferencing on RDF graphs in Autonomous Database. You can use a REST API, a SPARQL endpoint, a web UI to run SPARQL queries, and developer APIs for advanced RDF graph data management operations.

You can use these APIs by deploying Oracle RDF Graph Server and Query UI from OCI Marketplace.

See Using RDF Graph Server and Query UI for more information.
Oracle Spatial Database is included in Autonomous Database, allowing developers and analysts to get started easily with location intelligence analytics and mapping services.

About Oracle Spatial with Autonomous Database

Autonomous Database includes Oracle Spatial. Oracle Spatial allows developers and analysts to use spatial analysis in every application from basic spatial search and analysis to advanced geospatial applications and Geographic Information Systems (GIS). Organizations can manage different types of geospatial data, perform hundreds of spatial analytic operations, and use interactive map visualization tools with the spatial features in Oracle Autonomous Database and Oracle Database.

The spatial features provide a schema and functions that facilitate the storage, retrieval, update, and query of collections of spatial features in Autonomous Database. Spatial consists of the following:

• A schema (MDSYS) that prescribes the storage, syntax, and semantics of supported geometric data types
• A spatial indexing mechanism
• Operators, functions, and procedures for performing area-of-interest queries, spatial join queries, and other spatial analysis operations
• Utility functions and procedures for validating, loading, extracting, and working with spatial data
• GeoRaster, a feature that lets you store, index, query, analyze, and deliver GeoRaster data, that is, raster image and gridded data and its associated metadata

See Spatial Concepts and Oracle Database Spatial for more information.

Oracle Spatial Limitations with Autonomous Database

Autonomous Database includes Oracle Spatial, with some limitations.

You can use the Oracle spatial features with Autonomous Database, with the following limitations:

• Geocoding is available through Oracle Spatial Studio.
  See Oracle Spatial Studio for more information.
• The following Oracle Spatial components require that you download applications from Oracle Cloud Marketplace:
  – Oracle Spatial Studio
  – Oracle Spatial Map Visualization
– Network Data Model

• The following are not available in Autonomous Database:
  – Open Geospatial Consortium Web Services
  – Router
You may use the following methods to access cloud resources securely without storing user credentials: Oracle Cloud Infrastructure Resource Principals, AWS Amazon Resource Names (ARN)s, or Azure service principal.

Topics
- Use Resource Principal to Access Oracle Cloud Infrastructure Resources
- Use Amazon Resource Names (ARNs) to Access AWS Resources
- Use Azure Service Principal to Access Azure Resources

Use Resource Principal to Access Oracle Cloud Infrastructure Resources

You can use an Oracle Cloud Infrastructure resource principal with Autonomous Database. You or your tenancy administrator define the Oracle Cloud Infrastructure policies and a dynamic group that allows you to access Oracle Cloud Infrastructure resources with a resource principal. You do not need to create a credential object and Autonomous Database creates and secures the resource principal credentials you use to access the specified Oracle Cloud Infrastructure resources.

Topics
- About Using Resource Principal to Access Oracle Cloud Infrastructure Resources
- Perform Prerequisites to Use Resource Principal with Autonomous Database
- Enable Resource Principal to Access Oracle Cloud Infrastructure Resources
- Disable Resource Principal on Autonomous Database
- Use Resource Principal with DBMS_CLOUD

About Using Resource Principal to Access Oracle Cloud Infrastructure Resources

You can use a resource principal to authenticate and access Oracle Cloud Infrastructure resources. The resource principal consists of a temporary session token and secure credentials that enables the database to authenticate itself to other Oracle Cloud Infrastructure services. Using a resource principal to access services, the token stored with the credentials on Autonomous Database is only valid for the resources to which the dynamic group has been granted access.

To use Resource Principal, you or your tenancy administrator define the Oracle Cloud Infrastructure policies and a dynamic group that allows you to access Oracle Cloud Infrastructure resources.
Infrastructure resources with a resource principal. You do not need to create a credential object and Autonomous Database creates and secures the resource principal credentials you use to access the specified Oracle Cloud Infrastructure resources.

For example, while using Autonomous Database you might want to use Oracle Cloud Infrastructure resources to do the following:

- Access data from an Object Storage bucket, perform some operation on the data, and then write the modified data back to the Object Storage bucket.
- Access your vaults, keys, or secrets.
- List work requests or list work request errors.

When you are working with the database, you authenticate and access the database as a database user. An Autonomous Database user does not have an Oracle Cloud Infrastructure Identity and Access Management (IAM) identity, so as an Autonomous Database user you cannot use your database credentials to access Oracle Cloud Infrastructure services. Without a resource principal you must obtain credentials to access Oracle Cloud Infrastructure resources and create a credential object to access a resource from Autonomous Database.

A resource principal enables resources to be authorized to perform actions on Oracle Cloud Infrastructure services. Each resource has its own identity, and the resource authenticates using the certificates that are added to it. These certificates are automatically created, assigned to resources, and rotated, avoiding the need for you to create and manage your own credentials to access the resource.

Autonomous Database lets you use a resource principal to authenticate to Oracle Cloud Infrastructure APIs using the following interfaces:

- **DBMS_CLOUD** procedures and functions that take a credential argument
- **Oracle Cloud Infrastructure PL/SQL SDK APIs**

When you authenticate using a resource principal, Autonomous Database provides a secure method to access Oracle Cloud Infrastructure resources.

There are several steps required to set up a resource principal on Autonomous Database:

- You must create and define Oracle Cloud Infrastructure Identity and Access Management (IAM) policies. See Perform Prerequisites to Use Resource Principal with Autonomous Database for more information.
- You must enable the resource principal for the ADMIN user, and optionally enable the resource principal for a database user. See Enable Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

When you authenticate using a resource principal, you do not need to create and manage credentials to access Oracle Cloud Infrastructure resources. Autonomous Database makes the resource principal available to you and secures the resource principal for you.

### Perform Prerequisites to Use Resource Principal with Autonomous Database

Prior to making a call to an Oracle Cloud Infrastructure resource using a resource principal, an Oracle Cloud Infrastructure tenancy administrator must create Oracle...
Cloud Infrastructure policies, dynamic groups, and rules that define the resource principal privileges.

Perform the following steps before you use a resource principal with Autonomous Database:

1. Create an Oracle Cloud Infrastructure dynamic group.
   a. In the Oracle Cloud Infrastructure console click **Identity and Security** and click **Dynamic Groups**
   b. Click **Create Dynamic Group** and enter a **Name**, a **Description**, and a rule or use the **Rule Builder** to add a rule.
   c. Click **Create**.

   Resources that meet the rule criteria are members of the dynamic group. When you define a rule for a dynamic group, consider what resource is going to be given access to other resources.

   For example, consider the following examples:
   
   • Allow a specific Autonomous Database instance to access a resource.
     The Autonomous Database is specified in the `resource.id` parameter with an OCID:
     
     ```
     resource.id = '<your_Autonomous_Database_instance_OCID>'
     ```
   
   • Allow all Autonomous Databases in a compartment.
     The Autonomous Databases are specified in the `resource.type` parameter and the compartment is identified by a specified OCID in the `resource.compartment.id` parameter:
     
     ```
     ALL {resource.type = 'autonomousdatabase', resource.compartment.id = '<your_Compartment_OCID>'}
     ```
   
   • Allow all resources in the compartment
     The resource type identified by the OCID, specified in the `resource.compartment.id` parameter:
     
     ```
     ALL {resource.compartment.id='<your_Compartment_OCID>}'
     ```

   See [Managing Dynamic Groups](#) for more information on creating a dynamic group and creating rules to add resources to the group.

2. Write policy statements for the dynamic group to enable access to Oracle Cloud Infrastructure resources.
   a. In the Oracle Cloud Infrastructure console click **Identity and Security** and click **Policies**.
   b. To write policies for a dynamic group, click **Create Policy**, and enter a **Name** and a **Description**.
   c. Use the **Policy Builder** to create a policy.
For example to create a policy to allow access to Oracle Cloud Infrastructure Object Store to manage buckets and objects in the Object Store in a tenancy:

- Allow dynamic-group Example5 to manage buckets in tenancy
- Allow dynamic-group Example5 to manage objects in tenancy

d. Click **Create**.

See [Managing Policies](#) for more information on policies.

---

**Note:**
The resource principal token is cached for two hours. Therefore, if you change the policy or the dynamic group, you have to wait for two hours to see the effect of your changes.

---

### Enable Resource Principal to Access Oracle Cloud Infrastructure Resources

Perform the following steps to enable resource principal on Autonomous Database.

As a prerequisite, configure dynamic groups and policies. See [Perform Prerequisites to Use Resource Principal with Autonomous Database](#) for more information.

To enable a resource principal on Autonomous Database:

1. As the ADMIN user, enable resource principal for the Autonomous Database instance.

   For example:

   ```plsql
   EXEC DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL();
   ```

   PL/SQL procedure successfully completed.

   See [ENABLE_RESOURCE_PRINCIPAL Procedure](#) for more information.

   This creates the credential `OCI$RESOURCE_PRINCIPAL`.

2. (Optional) This step is only required if you want to grant access to the resource principal credential to a database user other than the ADMIN user. As the ADMIN user, enable resource principal for a specified database user.

   For example:

   ```plsql
   EXEC DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL(username => 'adb_user');
   ```

   PL/SQL procedure successfully completed.

   This grants the user `adb_user` access to the credential `OCI$RESOURCE_PRINCIPAL`.

   If you want the specified user to have privileges to enable resource principal for other users, set the `grant_option` parameter to `TRUE`.
For example:

```
BEGIN
DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL(
    username => 'adb_user',
    grant_option => TRUE);
END;
/
```

After you run this command, `adb_user` can enable resource principal for another user. For example, if you connect as `adb_user`, you can run the following command:

```
EXEC DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL(username => 'adb_user2');
```

See `ENABLE_RESOURCE_PRINCIPAL Procedure` for more information.

3. Verify that the resource principal credential is enabled.

For example, as the ADMIN user:

```
SELECT owner, credential_name FROM dba_credentials
    WHERE credential_name = 'OCI$RESOURCE_PRINCIPAL' AND owner = 'ADMIN';
```

<table>
<thead>
<tr>
<th>OWNER</th>
<th>CREDENTIAL_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMIN</td>
<td>OCI$RESOURCE_PRINCIPAL</td>
</tr>
</tbody>
</table>

For example, as a non-ADMIN user:

```
SELECT grantee, table_name, grantor, FROM ALL_TAB_PRIVS
    WHERE grantee = 'ADB_USER';
```

<table>
<thead>
<tr>
<th>GRANTEE</th>
<th>TABLE_NAME</th>
<th>GRANTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADB_USER</td>
<td>OCI$RESOURCE_PRINCIPAL</td>
<td>ADMIN</td>
</tr>
</tbody>
</table>

**Disable Resource Principal on Autonomous Database**

Shows the steps to disable resource principal for all Autonomous Database users or for a specified user.

1. To disable resource principal for all users, as the ADMIN user, run the following command:

```
EXEC DBMS_CLOUD_ADMIN.DISABLE_RESOURCE_PRINCIPAL();
```

   This removes the credential `OCI$RESOURCE_PRINCIPAL`.

2. Verify that the resource principal credential is disabled.
For example:

```
SELECT owner, credential_name FROM dba_credentials
    WHERE credential_name = 'OCI$RESOURCE_PRINCIPAL' AND owner = 'ADMIN';
```

No rows selected

To remove access to the resource principal credential for a specified database user, include the `username` parameter. This denies the specified user access to the `OCI$RESOURCE_PRINCIPAL` credential.

For example:

```
EXEC DBMS_CLOUD_ADMIN.DISABLE_RESOURCE_PRINCIPAL(username => 'ADB_USER');
```

See [DISABLE_RESOURCE_PRINCIPAL Procedure](#) for more information.

**Use Resource Principal with DBMS_CLOUD**

When you specify a resource principal credentials in `DBMS_CLOUD` calls, the database authenticates the Oracle Cloud Infrastructure requests for you and the database provides the credentials to access to the resources.

If you have not already done so, perform the required prerequisite steps:

- Access to Oracle Cloud Infrastructure resources depends on the dynamic group rules and the policies you set in Oracle Cloud Infrastructure policies and dynamic groups. See [Perform Prerequisites to Use Resource Principal with Autonomous Database](#) for more information.

- After you define the dynamic group and policies, enable the `ADMIN` schema or another schema to use a resource principal. See [Enable Resource Principal to Access Oracle Cloud Infrastructure Resources](#) for more information.

To use a `DBMS_CLOUD` procedure with resource principal credentials:

- Use a `DBMS_CLOUD` procedure or function and specify `OCI$RESOURCE_PRINCIPAL` as the credential name.

For example, you can access the Oracle Cloud Infrastructure Object Storage using resource principal credentials:

```
CREATE TABLE CHANNELS
    (channel_id CHAR(1),
     channel_desc VARCHAR2(20),
     channel_class VARCHAR2(20));
/
BEGIN
    DBMS_CLOUD.COPY_DATA(
        table_name => 'CHANNELS',
        credential_name => 'OCI$RESOURCE_PRINCIPAL',
        file_uri_list => 'https://objectstorage.us-
```
Use Amazon Resource Names (ARNs) to Access AWS Resources

You can use Amazon Resource Names (ARNs) to access AWS resources with Autonomous Database.

Topics

• About Using Amazon Resource Names (ARNs) to Access AWS Resources
• Perform Autonomous Database Prerequisites to Use Amazon ARNs
• Perform AWS Management Prerequisites to Use Amazon Resource Names (ARNs)
• Create Credentials with ARN Parameters to Access AWS Resources
• Update Credentials with ARN Parameters for AWS Resources

About Using Amazon Resource Names (ARNs) to Access AWS Resources

When you use ARN role based authentication with Autonomous Database, you can securely access AWS resources without creating and saving credentials based on long-term AWS IAM access keys.

For example, you may want to load data from an AWS S3 bucket into your Autonomous Database, perform some operation on the data, and then write the modified data back to the S3 bucket. You can do this without using an ARN if you have AWS user credentials to access the S3 bucket. However, using role-based ARNs to access AWS resources from Autonomous Database has the following benefits:

• You can create role-based access, with different policies for different users or schemas that need access to AWS resources from an Autonomous Database instance. This allows you to set a policy to limit access to AWS resources by role. For example, setting a policy limiting to read-only access, by role, to an S3 bucket.

• ARN based credentials provide better security as you do not need to provide long-term AWS user credentials in code to access AWS resources. Autonomous Database manages the temporary credentials generated from the AWS Assume Role Operation.

About Steps to Configure ARN Usage with Autonomous Database

Before creating a credential using an ARN in Autonomous Database, in AWS, your account administrator must define a policy that allows you to access AWS resources, such as an S3 bucket. By default, ARN credential services are not enabled on Autonomous Database. The
ADMIN user enables ARN credentials for the necessary user which allows them to create and use ARN credentials on the Autonomous Database instance.

In AWS, the role ARN is the identifier for the provided access and can be viewed on the AWS console. For added security, when the AWS administrator configures the role, policies, and trust relationship for the AWS account, they must also configure an “External ID” in the role’s trust relationship.

**Note:**

Setting the External ID is required for security.

The External ID provides additional protection for assuming roles. You configure the External ID as one of the following: the Autonomous Database compartment OCID, database OCID, or tenancy OCID. On AWS, the role can only be assumed by trusted users that are identified by the External ID included in the request URL, where the supplied External ID in the request matches the External ID configured in the role’s trust relationship.

The following figure outlines the configuration steps:

For details on the steps to configure Autonomous Database to access AWS resources, see the following:

- **Prerequisite steps on Autonomous Database:** On Autonomous Database you must enable the ADMIN user or another user to use credentials with ARN parameters to access AWS resources.
  
  See Perform Autonomous Database Prerequisites to Use Amazon ARNs for more information.

- **Prerequisite steps in the AWS Account:** In your AWS account, from the AWS Management Console or using the CLI, create the roles and policies for the ARN that you use with Autonomous Database, and update the trust relationship for the role. The Oracle user ARN is configured when the trust relationship for the role is updated.
  
  See Perform AWS Management Prerequisites to Use Amazon Resource Names (ARNs) for more information.
About Steps to Use ARNs with DBMS_CLOUD

Each AWS resource has its own identity, and the resource authenticates with the Autonomous Database instance using a DBMS_CLOUD credential that you create with parameters that identify the ARN. Autonomous Database creates and secures the principal credentials you use to access AWS resources.

To create a credential with ARN parameters to access AWS resources:

- Create credentials with DBMS_CLOUD.CREATE_CREDENTIAL and supply the parameters that identify an AWS role. Using the credential object, Autonomous Database can access AWS resources as specified in the policies defined for the role in the AWS account.
- Use the credential object you created in the previous step with a DBMS_CLOUD procedure or function that takes a credential parameter, such as DBMS_CLOUD.COPY_DATA or DBMS_CLOUD.LIST_OBJECTS.

See Create Credentials with ARN Parameters to Access AWS Resources for details on these steps.

Perform Autonomous Database Prerequisites to Use Amazon ARNs

Prior to using an AWS resource with DBMS_CLOUD.CREATE_CREDENTIAL with an ARN parameter, the ADMIN user must enable ARN on the Autonomous Database instance.

By default, ARN credential services are not enabled on Autonomous Database. The ADMIN user runs the procedure DBMS_CLOUD_ADMIN.ENABLE_AWS_ARN to enable the ADMIN user or other users to create credentials with ARN parameters.

1. Enable the use of ARN credentials on the Autonomous Database instance.

   BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_AWS_ARN(
     username => 'adb_user');
   END;
   /

   If you want the specified user to have privileges to enable ARN credentials for other users, set the grant_option parameter to TRUE.

   For example:

   BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_AWS_ARN(
     username => 'adb_user',
     grant_option => TRUE);
   END;
   /

   After you run this command, adb_user has privileges to enable ARN credentials for other users. For example, if you connect as adb_user, you can run the following command:

   BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_AWS_ARN(
username => 'adb_user2');
END;
/

See ENABLE_AWS_ARN Procedure for more information.

2. Query the CLOUD_INTEGRATIONS view to obtain Oracle's AWS user ARN.

```sql
SELECT param_value FROM CLOUD_INTEGRATIONS
WHERE param_name = 'aws_user_arn';
```

<table>
<thead>
<tr>
<th>PARAM_VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>arn:aws:iam::account-ID:user/username</td>
</tr>
</tbody>
</table>

The view CLOUD_INTEGRATIONS is available to the ADMIN user or to a user with DWROLE privileges.

The AWS administrator uses the aws_user_arn value when configuring the AWS role's trust relationship with the role and policies on the AWS system. Providing this value grants permission on the AWS side for DBMS_CLOUD to access AWS resources.

Perform AWS Management Prerequisites to Use Amazon Resource Names (ARNs)

Using the AWS Management Console or using the APIs, create an AWS user, role, policies, and trust relationship. You perform these steps before you use with DBMS_CLOUD.CREATE_CREDENTIAL to create a credential with an ARN parameter on Autonomous Database.

To use an ARN to access AWS resources your AWS administrator defines the policies and a principal that allows you to access AWS resources. For example, while using Autonomous Database you might want to access data from an S3 bucket, perform some operation on the data, and then write the modified data back to the S3 bucket.

Note:

Depending on your existing AWS configuration and the External ID you use, you do not need to create a new role and policy for each Autonomous Database instance. If you already have an AWS role containing the necessary policy to access a resource, for example to access S3 cloud storage, you can modify the trust relationship to include the details in Step 3. Likewise, if you already have a role with the necessary trust relationship, you can use that role to access all of your databases in an OCI compartment or tenancy if you use an external ID that specifies the compartment OCID or tenancy OCID.

From the AWS Management Console or using the APIs, an AWS administrator performs the following steps:
1. Create a policy. In the policy you specify permissions for accessing AWS resources such as S3 buckets.

   See Creating an IAM policy to access Amazon S3 resources for more information.

2. Create a role and attach the policy to the role.

   a. Access the AWS Management Console and choose Identity and Access Management (IAM).

   b. Click **Create role**.

   ![IAM Dashboard](image)

   c. Select **Another AWS account**.

   ![Create Role](image)

   d. Enter your **Account ID**.

      You use this as a temporary value. Later you replace this with the Account ID you use to access AWS resources.

   e. In the **Options** area select **Require external ID** and enter a temporary external ID, such as 0000. Later you replace this external ID with a valid value.

   f. Click **Next Permissions** to attach the Policies you created in Step 1 or other policies you want to apply to the role.

   g. Click **Next Tags** and apply or create tags as needed for the role.

   h. Click **Next Review** and add a **Role Name** and **Role Description**.

   i. Click **Create Role**.
You use the role’s ARN with `DBMS_CLOUD.CREATE_CREDENTIAL` to create credential objects with ARN parameters to access AWS resources.

See [Creating a role to delegate permissions to an IAM user](#) for more information.

3. Specify a Trust Relationship for the role.
   a. From the Roles list, under **Role name**, select the role you created.
   b. On the roles Summary page for the selected role, select the **Trust relationships** tab.
   c. In the trust relationship, click **Edit trust relationship**.
   d. Edit the trust relationship to specify the Principal parameter **AWS**.
      This AWS user ARN is available in the CLOUD_INTEGRATIONS view. See [Perform Autonomous Database Prerequisites to Use Amazon ARNs](#) for more information.
   e. Edit the trust relationship to specify the External ID.
      On Autonomous Database when you create an AWS ARN credential with `DBMS_CLOUD.CREATE_CREDENTIAL`, by default the **external_id_type** parameter value is **database_ocid**. Optionally you can set the **external_id_type** type as one of: **database_ocid**, **compartment_ocid**, or **tenant_ocid**.
      When you use the database OCID as the External ID, the policy’s trust relationship only trusts the Autonomous Database instance specified with the OCID. If you use a compartment OCID, the policy’s trust relationship trusts all
the Autonomous Database instances in the compartment and you can use the same role ARN to grant access to AWS resources to any Autonomous Database in the specified compartment. Likewise, if you use the tenancy OCID, you can use the same role ARN to grant access to AWS resources to any Autonomous Database in the specified tenancy.

Previously in Step 2 you set the trust relationship External ID to the temporary value 0000.

On AWS you configure the trust relationship External ID value to match one of the following:

- **When the** `external_id_type` **type is** `database_ocid`, **on AWS you configure the role's trust relationship External ID to be the Database OCID.**

  The Database OCID is available by running the following query:

  ```sql
  SELECT cloud_identity FROM v$pdb;
  ```

  See [Obtain Tenancy Details](#) for more information.

- **When the** `external_id_type` **type is** `compartment_ocid`, **on AWS you configure the role's trust relationship External ID to be the Compartment OCID.**

  The Compartment OCID is available on the Compartment details page from the Oracle Cloud Infrastructure Console. To find the Compartment details page, from the Oracle Cloud Infrastructure left navigation menu click **Identity & Security** and then select **Compartments**. Select the compartment that contains the Autonomous Database instance to see the Compartment ID.

- **When the** `external_id_type` **type is** `tenant_ocid`, **on AWS you configure the role's trust relationship External ID to be the Tenancy OCID.**

  The Tenancy OCID is available on the Tenancy details page from the Oracle Cloud Infrastructure Console. To find the Tenancy details page, from the Oracle Cloud Infrastructure left navigation menu click **Governance & Administration** and then select **Tenancy Details**. The Tenancy Information tab shows the Tenancy OCID.

- When you set the value for ExternaID, by default the OCID value must be in upper case. If you want to supply the OCID in lower case, set the condition "StringEqualsIgnoreCase" instead of "StringEquals" in the JSON when you edit the trust relationship.

  ![Edit Trust Relationship](image)

  See [How to use trust policies with IAM role](#) for more information.
After the ARN role configuration is finished, you can create a `DBMS_CLOUD.CREATE_CREDENTIAL` credential with ARN parameters to access Amazon resources from Autonomous Database. See Create Credentials with ARN Parameters to Access AWS Resources for more information.

Create Credentials with ARN Parameters to Access AWS Resources

After ARN usage is enabled for the Autonomous Database instance and the ARN is configured by the AWS administrator, on Autonomous Database you can create a credential object with ARN parameters. Autonomous Database creates and secures the principal credentials you use to access the Amazon resources when you supply the credential object with `DBMS_CLOUD` procedures and functions.

To use Amazon resources with Autonomous Database, do the following:

1. Create credentials using the procedure `DBMS_CLOUD.CREATE_CREDENTIAL` with the `params` parameter to specify the ARN value. For example:

   ```sql
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DEF_CRED_ARN',
       params =>
         JSON_OBJECT('aws_role_arn' value 'arn:aws:iam::123456:role/AWS_ROLE_ARN',
                     'external_id_type' value 'database_ocid')
     );
   END;
   /
   ``

   This operation creates the credentials in the database in an encrypted format. You can use any name for the credential name.

   For detailed information about the parameters, see `CREATE_CREDENTIAL` Procedure.

2. Use a `DBMS_CLOUD` procedure to access an Amazon resource with the ARN credentials.

   For example, use `DBMS_CLOUD.LIST_OBJECTS`.

   ```sql
   SELECT object_name FROM DBMS_CLOUD.LIST_OBJECTS(
     credential_name => 'DEF_CRED_ARN',
     location_uri => 'https://my-bucket.s3.us-west-2.amazonaws.com/');
   ```

Update Credentials with ARN Parameters for AWS Resources

The ARN credentials you use on Autonomous Database work with the AWS token service that enables you to use temporary role based credentials to access to AWS resources from Autonomous Database. Therefore, if an AWS Administrator revokes the policies, roles, or trust relationship, you need to either update the credentials or create new credentials to access the AWS resources.

Perform the following steps to update credentials:
1. Use `DBMS_CLOUD.UPDATE_CREDENTIAL` to update an ARN based credential to supply a new ARN value.

```sql
BEGIN
    DBMS_CLOUD.UPDATE_CREDENTIAL(
        credential_name => 'DEF_CRED_ARN',
        attribute => 'aws_role_arn',
        value => 'new_ARN_value');
END;
/
```

This updates the `aws_role_arn` attribute to the new value `new_ARN_value` for the credential named `DEF_CRED_ARN`.

2. Use `DBMS_CLOUD.UPDATE_CREDENTIAL` to update an ARN based credential to update the attribute `external_id_type` value.

```sql
BEGIN
    DBMS_CLOUD.UPDATE_CREDENTIAL(
        credential_name => 'DEF_CRED_ARN',
        attribute => 'external_id_type',
        value => 'compartment_ocid');
END;
/
```

This updates the `external_id_type` attribute value to the value `compartment_ocid`.

See `UPDATE_CREDENTIAL Procedure` and `CREATE_CREDENTIAL Procedure` for more information.

Use Azure Service Principal to Access Azure Resources

You can use an Azure service principal with Autonomous Database to access Azure resources without having to create and store your own credential objects in the database.

Topics
- Enable Azure Service Principal
- Provide Azure Application Consent and Assign Roles
- Use Azure Service Principal with DBMS_CLOUD
- Disable Azure Service Principal
- Notes for Azure Service Principal
Enable Azure Service Principal

Enable Azure service principal authentication to allow Autonomous Database to access Azure services without providing long-term credentials.

**Note:**
To use Autonomous Database with Azure service principal authentication you need a Microsoft Azure account. See Microsoft Azure for details.

To enable Azure service principal authentication on Autonomous Database:

1. Obtain your Microsoft Azure Active Directory tenant ID.  
   See How to find your Azure Active Directory tenant ID for more information.

2. Enable Azure service principal with `DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH`.
   For example:

   ```sql
   BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
   provider => 'AZURE',
   username => 'adb_user',
   params   => JSON_OBJECT('azure_tenantid' value
   'azure_tenaidid'));
   END;
   /
   ```

   This enables Azure service principal authentication and creates an Azure application on Autonomous Database.

   If you want the specified user to have privileges to enable Azure service principal for other users, set the `params` parameter `grant_option` to `TRUE`.

   For example:

   ```sql
   BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
   provider => 'AZURE',
   username => 'adb_user',
   params   => JSON_OBJECT('grant_option' value TRUE,
   'azure_tenantid' value 'azure_tenaidid'));
   END;
   /
   ```

   After you run this command, `adb_user` can enable Azure service principal for another user. For example, if you connect as `adb_user`, you can run the following command:

   ```sql
   BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
   provider => 'AZURE',
   username =>
   `adb_user',
   params   => JSON_OBJECT('grant_option' value TRUE,
   'azure_tenantid' value 'azure_tenaidid');
   END;
   ```
Provide Azure Application Consent and Assign Roles

To access Azure resources from Autonomous Database with Azure service principal authentication you must consent the Azure application and assign roles to allow access to your Azure resources.

To provide Azure application consent and assign roles, perform the following steps:

1. On Autonomous Database query CLOUD_INTEGRATIONS.

   For example:

   ```
   SELECT param_name, param_value FROM CLOUD_INTEGRATIONS;
   ```

<table>
<thead>
<tr>
<th>PARAM_NAME</th>
<th>PARAM_VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>azure_tenantid</td>
<td>29981886-6fb3-44e3-82ab-d870b0e8e7eb</td>
</tr>
<tr>
<td>azure_consent_url</td>
<td><a href="https://login.microsoftonline.com/f8cdef31-91255a/oauth2/v2.0/authorize?client_id=d66f1b5-1250d5445c0b&amp;response_type=code&amp;scope=User.read">https://login.microsoftonline.com/f8cdef31-91255a/oauth2/v2.0/authorize?client_id=d66f1b5-1250d5445c0b&amp;response_type=code&amp;scope=User.read</a></td>
</tr>
<tr>
<td>azure_app_name</td>
<td>ADBS_APP_OCID1.AUTONOMOUSDATABASE.REGION1.SEA.ANZWKLJS2LYNB3AAWLYL3JVC4ICEXLB3ZG6WTCX735JSSY2NRHOBU4DZO0VA</td>
</tr>
</tbody>
</table>

   The view CLOUD_INTEGRATIONS is available to the ADMIN user or to a user with DWROLE role.

2. In a browser, open the Azure consent URL, specified by the azure_consent_url parameter.

   For example, copy and enter the URL in your browser:

   ```
   https://login.microsoftonline.com/f8cdef31-91255a/oauth2/v2.0/authorize?client_id=d66f1b5-1250d5445c0b&response_type=code&scope=User.read
   ```

   The Permissions requested page opens and shows a consent request, similar to the following:
3. To provide consent click **Accept**.

4. On the Microsoft Azure console, assign the roles you want to grant to allow access to the specified Azure resources.

   For example, if you want to access Azure Blob Storage from Autonomous Database, assign roles so that the Azure application (the service principal) has access to Azure Blob Storage.

   **Note:**
   
   To work with Azure Blob Storage, you need an Azure storage account. If you do not have an Azure storage account, create a storage account. See [Create a storage account](#) for more information.
a. On the Microsoft Azure console, under **Azure services**, select **Storage accounts**.

b. Under **Storage accounts**, click the storage account you want to grant service principal access to.

c. On the left, click **Access Control (IAM)**.

d. From the top area, click **+ Add → Add role assignment**.

e. In the search area enter text to narrow the list of roles that you see. For example, enter **Storage Blob** to show the available roles that contain **Storage Blob**.

f. Select one or more roles as appropriate for the access you want to grant. For example, select **Storage Blob Data Contributor**.

g. Click **Next**.

h. In the **Add role assignment**, under **Members** click **+ Select members**.

i. Under **Select members**, in the select field, enter the **azure_app_name** listed in Step 1 (the **param_value** column of the **CLOUD_INTEGRATIONS** view).

j. Select the application.

   For example, click
   
   ADBS_APP_OCID1.AUTONOMOUSDATABASE.REGION1.SEA.ANZWKLJSZLYNB3AAWLYL3JVC4IC
   EXLB3ZG6WTCX735JSSY2NRH0BU4DZ00VA

k. Click **Select**.

l. Click **Review + assign**.

5. **Click Review + Assign** again.

After assigning a role you need to wait, as role assignments may take up to five minutes to propagate in Azure.

This example shows steps to grant roles for accessing Azure Blob Storage. If you want to provide access for other Azure services you need to perform equivalent steps for the additional Azure services to allow the Azure application (the service principal) to access the Azure service.

**Use Azure Service Principal with DBMS_CLOUD**

When you make **DBMS_CLOUD** calls to access Azure resources and specify the credential name as **AZURE$PA**, the authentication on the Azure side happens using the Azure service principal.

If you have not already done so, perform the required prerequisite steps:

- Enable the **ADMIN** schema or another schema to use Azure service principal authentication. See **Enable Azure Service Principal** for more information.

- Consent the application and perform the Azure role assignment grants. See **Provide Azure Application Consent and Assign Roles** for more information.

To use a **DBMS_CLOUD** procedure or function with Azure service principal, specify **AZURE$PA** as the credential name. For example, you can access Azure Blob Storage using Azure service principal credentials as follows:

```
SELECT * FROM DBMS_CLOUD.LIST_OBJECTS('AZURE$PA', 'https://treedata.blob.core.windows.net/treetypes/');
```
If you compare the steps required to access object storage, as shown in Create Credentials and Copy Data into an Existing Table, notice that Step 1, creating credentials is not required because you are using an Azure service principal called AZURE$PA.

**Disable Azure Service Principal**

To disable access to Azure resources from Autonomous Database with Azure service principal, use `DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH`.

To disable Azure service principal on Autonomous Database:

```sql
BEGIN
    DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH(
        provider => 'AZURE',
        username => 'adb_user');
END;
/
```

When the `provider` value is `AZURE` and the `username` is a user other than the `ADMIN` user, the procedure revokes the privileges from the specified user. In this case, the `ADMIN` user and other users can continue to use `ADMIN.AZURE$PA` and the application that is created for the Autonomous Database instance remains on the instance.

When the `provider` value is `AZURE` and the `username` is `ADMIN`, the procedure disables Azure service principal based authentication and deletes the Azure service principal application on the Autonomous Database instance. In this case, if you want to enable Azure service principal you must perform all the steps required to use Azure service principal again, including the following:

- Enable the `ADMIN` schema or another schema to use Azure service principal authentication. See Enable Azure Service Principal for more information.
- Consent the application and perform the Azure role assignment grants. See Provide Azure Application Consent and Assign Roles for more information.

See DISABLE_PRINCIPAL_AUTH Procedure for more information.

**Notes for Azure Service Principal**

Notes for using Azure service principal.

- **Cloning an Autonomous Database instance with Azure service principal:** When you clone an instance with Azure service principal enabled, the Azure service principal configuration is not carried over to the clone. Perform the steps to enable Azure service principal on the clone if you want to enable Azure service principal on a cloned instance.
Using Application Continuity on Autonomous Database

Autonomous Database provides application continuity features for making connections to the database.

Topics

- About Application Continuity on Autonomous Database
- Configure Application Continuity on Autonomous Database
- Client Configuration for Continuous Availability on Autonomous Database
- Notes for Application Continuity on Autonomous Database

About Application Continuity on Autonomous Database

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

Your applications achieve continuous availability when planned maintenance, unplanned outages, and load rebalances of the database are hidden from the application. The combination of application coding best practices, application continuity configuration, and Autonomous Database ensures that your applications are continuously available.

The best approach for hiding planned maintenance activities from your applications is to transparently drain or failover applications. Oracle’s connection pools and mid-tiers, including the WebLogic Server, Oracle Universal Connection Pool (UCP), OCI session pool and ODP.NET Unmanaged Provider are Fast Application Notification (FAN) aware and therefore are notified when maintenance is underway on Autonomous Database to allow graceful draining of work before maintenance. Application Continuity runs during planned maintenance to failover those sessions that do not drain in the predefined drain interval (5 minutes on Autonomous Database).

In order to hide unplanned outages resulting from a component or communication failure Oracle provides:

- **Notification.** FAN is the first step to hiding outages. FAN notifies clients and breaks them out of their current network wait when an outage occurs. This avoids stalling applications for long network waits. For Autonomous Database, FAN is handled at the driver and by the Autonomous Database cloud connection manager.

  FAN notification automatically triggers closing idle connections, opening new connections in the new service location, and allows a configurable time for active work to complete in the soon-to-be-shutdown service location. The major third-party JDBC mid-tiers, such as IBM WebSphere, allow the same behavior when configured with UCP. For JDBC-based applications that cannot use UCP, Oracle provides solutions using Oracle Drivers...
and connection tests. On Autonomous Database FAN for planned maintenance is sent in-Band.

- **Recovery.** After the client is notified, failover handling with Transparent Application Continuity (TAC) or Application Continuity (AC) re-establishes a connection to the Autonomous Database and replays in-flight, uncommitted, work when possible. By replaying in-flight work, the application can usually continue executing without knowing that any failure happened.

You enable Application Continuity on Autonomous Database in one of two configurations, depending on the application:

- **Application Continuity (AC)**
  Application Continuity hides outages for thin Java-based applications, and Oracle Database Oracle Call Interface and ODP.NET based applications with support for open-source drivers, such as Node.js, and Python. Application Continuity rebuilds the session by recovering the session from a known point which includes session states and transactional states. Application Continuity rebuilds all in-flight work. The application continues as it was, seeing a slightly delayed execution time when a failover occurs.

- **Transparent Application Continuity (TAC)**
  Transparent Application Continuity (TAC) transparently tracks and records session and transactional state so the database session can be recovered following recoverable outages. This is done with no reliance on application knowledge or application code changes, allowing Transparent Application Continuity to be enabled for your applications. Application transparency and failover are achieved by consuming the state-tracking information that captures and categorizes the session state usage as the application issues user calls.

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

**Note:**

By default Application Continuity is disabled on Autonomous Database.

**Configure Application Continuity on Autonomous Database**

To configure Application Continuity you must enable application continuity for the database service your application uses and configure the failover type and the drain timeout. In addition, you must set several connection string parameters that enable high availability.

**Topics**

- Configure Your Service to Enable Application Continuity
- Use Fast Application Notification (FAN)
- Configure Connection String for High Availability
- Configure Driver Specific Client Options
Configure Your Service to Enable Application Continuity

Use `DBMS_APP_CONT_ADMIN` to enable Application Continuity or Transparent Application Continuity:

- **Application Continuity (AC):** Set this failover option using the procedure `DBMS_APP_CONT_ADMIN.ENABLE_AC`. The `ENABLE_AC` procedure takes three parameters: `SERVICE_NAME` is the service name to change, `FAILOVER_RESTORE`, set to `LEVEL1` to select Application Continuity (AC), and `REPLAY_INITIATION_TIMEOUT`, is the replay timeout that specifies how many seconds after a request is submitted to allow that request to replay.

  For example, as the ADMIN user, to enable Application Continuity for the `TPURGENT` service:

  ```sql
  execute DBMS_APP_CONT_ADMIN.ENABLE_AC(
    'databaseid_tpurgent.adb.oraclecloud.com', 'LEVEL1', 600);
  ```

- **Transparent Application Continuity (TAC):** Set this failover option using the procedure `DBMS_APP_CONT_ADMIN.ENABLE_TAC`. The `ENABLE_TAC` procedure takes three parameters: `SERVICE_NAME` is the service name to change, `FAILOVER_RESTORE`, set to `AUTO` to select Transparent Application Continuity (TAC), and `REPLAY_INITIATION_TIMEOUT` is the replay timeout that specifies how many seconds after a request is submitted to allow that request to replay.

  For example, as the ADMIN user, to enable Transparent Application Continuity for the `TP` service with the replay timeout set to 20 minutes:

  ```sql
  execute DBMS_APP_CONT_ADMIN.ENABLE_TAC(
    'databaseid_tp.adb.oraclecloud.com', 'AUTO', 1200);
  ```

- **Disabled:** Disable failover using the procedure `DBMS_APP_CONT_ADMIN.DISABLE_FAILOVER()`.

  For example, as the ADMIN user, to disable failover for the `TP` service:

  ```sql
  execute DBMS_APP_CONT_ADMIN.DISABLE_FAILOVER(
    'databaseid_tp.adb.oraclecloud.com');
  ```

Find the Service Name Parameter for 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_adb1_high.adb.oraclecloud.com
  ```

- Depending on your workload type, use a command similar to the following to `SELECT` from `DBA_SERVICES` on your database and identify the service where you want to enable Application Continuity:
## Data Warehouse

```
SELECT name, failover_type FROM DBA_SERVICES;
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>FAILOVER_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvt21_adb1_low.adb.oraclecloud.com</td>
<td></td>
</tr>
<tr>
<td>nvt21_adb1_high.adb.oraclecloud.com</td>
<td></td>
</tr>
<tr>
<td>nvt21_adb1_medium.adb.oraclecloud.com</td>
<td></td>
</tr>
</tbody>
</table>

## Transaction Processing or JSON Database

```
SELECT name, failover_type FROM DBA_SERVICES;
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>FAILOVER_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvt21_adb1_tp.adb.oraclecloud.com</td>
<td></td>
</tr>
<tr>
<td>nvt21_adb1_tpurgent.adb.oraclecloud.com</td>
<td></td>
</tr>
<tr>
<td>nvt21_adb1_low.adb.oraclecloud.com</td>
<td>AUTO</td>
</tr>
<tr>
<td>nvt21_adb1_high.adb.oraclecloud.com</td>
<td>AUTO</td>
</tr>
<tr>
<td>nvt21_adb1_medium.adb.oraclecloud.com</td>
<td>AUTO</td>
</tr>
</tbody>
</table>

Notice the FAILOVER_TYPE for the high service has the no value and indicates that Application Continuity is disabled.

### Verify Application Continuity is Enabled for a Service

Depending on your workload type, check the output of the query on DBA_SERVICES to verify that Application Continuity is enabled.

### Data Warehouse

```
SELECT name, failover_type FROM DBA_SERVICES;
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>FAILOVER_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvt21_adb1_low.adb.oraclecloud.com</td>
<td></td>
</tr>
<tr>
<td>nvt21_adb1_high.adb.oraclecloud.com</td>
<td>AUTO</td>
</tr>
<tr>
<td>nvt21_adb1_medium.adb.oraclecloud.com</td>
<td>AUTO</td>
</tr>
</tbody>
</table>

### Transaction Processing or JSON Database

```
SELECT name, failover_type FROM DBA_SERVICES;
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>FAILOVER_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvt21_adb1_tp.adb.oraclecloud.com</td>
<td>TRANSACTION</td>
</tr>
<tr>
<td>nvt21_adb1_tpurgent.adb.oraclecloud.com</td>
<td>AUTO</td>
</tr>
<tr>
<td>nvt21_adb1_low.adb.oraclecloud.com</td>
<td>AUTO</td>
</tr>
<tr>
<td>nvt21_adb1_medium.adb.oraclecloud.com</td>
<td>AUTO</td>
</tr>
</tbody>
</table>

The FAILOVER_TYPE value for the high service is now AUTO, indicating that Transparent Application Continuity (TAC) is enabled and the FAILOVER_TYPE value for the tpurgent service is now TRANSACTION, indicating that Application Continuity (AC) is enabled.

---

Chapter 17
Configure Application Continuity on Autonomous Database
17-4
Use Fast Application Notification (FAN)

When connecting to Autonomous Database, the Oracle database auto-configures FAN. For application deployments with Autonomous Database, Fast Application Notification (FAN) events for unplanned outages are directed to the connection manager (CMAN) and no client application configuration steps required to use FAN.

FAN is automatically handled for you by the client driver and by the Autonomous Database Connection Manager (CMAN):

- For planned maintenance events, FAN is sent in-band, directly to the drivers. This requires that applications use Oracle Pools or TAC for request boundaries, or use connection tests.
- The Oracle Database and Oracle client drivers drain on connection tests and at request boundaries.

See Client Configuration for Continuous Availability on Autonomous Database for more information.

Configure Connection String for High Availability

To maintain high availability, Oracle recommends that you set certain connection string parameters when you connect to Oracle Autonomous Database.

Set the `CONNECT_TIMEOUT`, `RETRY_DELAY`, `RETRY_COUNT`, and `TRANSPORT_CONNECT_TIMEOUT` parameters in the connection string when you connect to Oracle Autonomous Database. The connect strings embedded in the Oracle-supplied `tnsnames.ora` file are preconfigured with appropriate values for most applications. In some cases, depending on your applications needs, you may need to change the preconfigured values for a connection string.

Use this TNS for all Oracle clients version 12.2 or higher:

```plaintext
alias =
(DESCRIPTION =
(CONNECT_TIMEOUT= 90)(RETRY_COUNT=50)
(RETRY_DELAY=3)(TRANSPORT_CONNECT_TIMEOUT=3)
(ADDRESS_LIST =
  (LOAD_BALANCE=on)
  (ADDRESS = (PROTOCOL = TCP)(HOST=scan-host)(PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME = service-name)))
```

Use the following for JDBC connections using Oracle driver version 12.1 or earlier:

```plaintext
alias =
(DESCRIPTION =
(CONNECT_TIMEOUT= 15)(RETRY_COUNT=50)
(RETRY_DELAY=3)
(ADDRESS_LIST =
  (LOAD_BALANCE=on)
  (ADDRESS = (PROTOCOL = TCP)(HOST=scan-host)(PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME = service-name)))
```

Notes for connection strings:

- For JDBC and ODP clients, the pool connection wait time should be configured to be longer than the `CONNECT_TIMEOUT` in the connect string.
Do not use Easy Connect Naming on the client because such connections do not have high-availability capabilities. See Download Client Credentials (Wallets) for information on the tnsnames.ora file.

Configure Driver Specific Client Options

Depending on your client and your driver, you need to assure that the client is properly configured to use Application Continuity when you connect to Autonomous Database.

Topics

- Configure JDBC Thin Driver
- Configure Oracle Call Interface (OCI) Driver
- Configure ODP.NET Unmanaged Provider Driver

Configure JDBC Thin Driver

Shows details to use Application Continuity with Autonomous Database with a client using the JDBC Thin Driver.

If your application uses the JDBC Thin Driver, follow these recommended practices:

1. Use JDBC Statement Cache for Coverage and Performance.

   For best coverage and performance, use the JDBC driver statement cache in place of an application server statement cache. This allows the driver to know that statements are closed and memory is to be freed at the end of requests.

   To use the JDBC statement cache, use connection property
   
   ```
   oracle.jdbc.implicitStatementCacheSize
   (OracleConnection.CONNECTION_PROPERTY_IMPLICIT_STATEMENT_CACHE_SIZE).
   
   The statement cache is per connection. The value for the cache size matches your number of open cursors. For example:
   
   ```
   
   ```
   oracle.jdbc.implicitStatementCacheSize=nnn
   where nnn is typically between 10 and 100 and is equal to the number of open cursors your application maintains.
   ```

2. Tune the Garbage Collector.

   For many applications the default Garbage Collector tuning is sufficient. For applications that return and keep large amounts of data you can use higher values, such as 2GB or larger. For example:

   ```
   java -Xms3072m -Xmx3072m
   ```

   It is recommended to set the memory allocation for the initial Java heap size (ms) and maximum heap size (mx) to the same value. This prevents using system resources on growing and shrinking the memory heap.

3. When using the Universal Connection Pool (UCP), disable Fast Connection Failover. For example:

   ```
   PoolDataSource.setFastConnectionFailoverEnabled(false)
   ```
Configure Oracle Call Interface (OCI) Driver

Shows details to use Application Continuity with Autonomous Database with a client using the Oracle Call Interface (OCI) driver.

If the client application uses the Oracle Call Interface (OCI) Driver, follow this recommended practice:

- Replace `OCIStmtPrepare` with `OCIStmtPrepare2`. `OCIStmtPrepare()` has been deprecated since 12.2. All applications should use `OCIStmtPrepare2()`. Transparent Application Continuity (TAC) and Application Continuity (AC) allow `OCIStmtPrepare` but do not replay this statement.

Do not configure ONS servers in `oraaccess.xml`:

```
<ons>
  <servers>
    <!--Do not enter any values -->
  </servers>
</ons>
```

Also, for Autonomous Database on shared Exadata infrastructure, do not configure the `<fan>` section:

```
<fan>
  <!-- only possible values are "trace" or "error" -->
  <subscription_failure_action>
  </subscription_failure_action>
</fan>
```

Configure ODP.NET Unmanaged Provider Driver

Shows details to use Application Continuity with Autonomous Database with a client using the ODP.NET Unmanaged Provider Driver.

The ODP.NET Unmanaged Provider driver automatically uses Application Continuity when Application Continuity is enabled on the database service that your application uses to connect to Autonomous Database.

When connecting an ODP.NET application to your Autonomous Database on shared Exadata infrastructure, do not configure ONS servers in `oraaccess.xml`:

```
<ons>
  <servers>
    <!--Do not enter any values -->
  </servers>
</ons>
```
Client Configuration for Continuous Availability on Autonomous Database

You do not need to restart applications for planned maintenance activities when you enable Application Continuity and you follow the coding best practices.

Topics
- Connect Using Database Services with Application Continuity Enabled
- Use Recommended Practices that Support Draining
- Steps for Using Application Continuity
- Developer Best Practices for Continuous Availability

Connect Using Database Services with Application Continuity Enabled

Oracle database services provide transparency for the underlying Autonomous Database infrastructure.

The high availability and application continuity operations are predicated on the use of Autonomous Database connection services. To obtain application continuity, use a database service when you connect to your database.

The names of the predefined database services on Autonomous Database are different, depending on your workload, as described in Database Service Names for Autonomous Data Warehouse and Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database.

Use Recommended Practices that Support Draining

On Autonomous Database there is never a need to restart application servers when planned maintenance follows best practice.

For planned maintenance, the recommended approach is to provide time for current work to complete before maintenance is started. On Autonomous Database this happens automatically and work is drained before starting maintenance activities when you follow these guidelines:

- FAN with Oracle Connection Pools or Oracle Drivers
- Connection tests

Use draining in combination with your chosen failover solution for those requests that do not complete within the allocated time for draining. Your failover solution will try to recover sessions that did not drain in the allocated time.

Return Connections to the Connection Pool

The application should return the connection to the connection pool on each request. It is best practice that an application checks-out a connection only for the time that it needs it. Holding a connection instead of returning it to the pool does not perform. An application should therefore check-out a connection and then check-in that connection immediately the work is complete. The connections are then available for later use by
other threads, or your thread when needed again. Returning connections to a connection pool is a general recommendation regardless of whether you use FAN to drain, or connection tests to drain.

**Use an Oracle Connection Pool**

Using a FAN-aware, Oracle connection pool is the recommended solution for hiding planned maintenance. As the maintenance progresses and completes, sessions are moved and rebalanced. There is no impact to users when your application uses an Oracle Pool with FAN and returns connections to the pool between requests. Supported Oracle Pools include UCP, WebLogic GridLink, Tuxedo, OCI Session Pool, and ODP.NET Managed and Unmanaged providers. No application changes whatsoever are needed to use FAN other than making sure that your connections are returned to pool between requests.

**Use UCP with a Third-Party Connection Pool**

If you are using a third party, Java-based application server, the most effective method to achieve draining and failover is to replace the pooled data source with UCP. This approach is supported by many application servers including Oracle WebLogic Server, IBM WebSphere, IBM Liberty, Apache Tomcat, Red Hat WildFly (JBoss), Spring, and Hibernate, and others.

**Use Connection Tests**

If you cannot use an Oracle Pool with FAN, then the Autonomous Database or provided client drivers will drain the session. When services are relocated or stopped during maintenance, or there is a switchover to a standby site using Autonomous Data Guard, the Oracle Database and Oracle client drivers look for safe places to release connections according to the following rules:

- Standard connection tests for connection validity at borrow or return from a connection pool
- Custom SQL tests for connection validity
- Request boundaries are in effect and the current request has ended

**Use Connection Tests with Autonomous Database**

You can add, delete, enable or disable connection tests for Autonomous Database.

Use the view `DBA_CONNECTION_TESTS` to show the available connection tests.

For example:

```sql
SQL> EXECUTE
dbms_app_cont_admin.add_sql_connection_test('SELECT COUNT(1) FROM DUAL');
SQL> EXECUTE
dbms_app_cont_admin.enable_connection_test(dbms_app_cont_admin.sql_test,
                                          'SELECT COUNT(1) FROM DUAL');
SQL> SELECT * FROM DBA_CONNECTION_TESTS;
```

Configure the same connection test that is enabled in your database at your connection pool or application server. Also configure flushing and destroying the pool on connection test failure to at least two times the maximum pool size or MAXINT.

**Use Connection Tests with Thin Java Driver**
If you would like to use connection tests that are local to the driver and cannot use UCP’s full FAN support:

- Enable validate-on-borrow=true
- Set the Java system properties:
  - -Doracle.jdbc.fanEnabled=true
  - -Doracle.jdbc.defaultConnectionValidation=SOCKET

And then use one of the following tests:

- java.sql.Connection.isValid(int timeout)
- oracle.jdbc.OracleConnection.pingDatabase()
- oracle.jdbc.OracleConnection.pingDatabase(int timeout)
- A HINT at the start of your test SQL:
  - /*+ CLIENT_CONNECTION_VALIDATION */

**Use Connection Tests with OCI Driver**

If you would like to use the OCI driver directly, use OCI_ATTR_SERVER_STATUS. This is the only method that is a code change. In your code, check the server handle when borrowing and returning connections to see if the session is disconnected. During maintenance, the value of OCI_ATTR_SERVER_STATUS is set to OCI_SERVER_NOT_CONNECTED. When using OCI session pool, this connection check is done for you.

The following code sample shows how to use OCI_ATTR_SERVER_STATUS:

```c
ub4 serverStatus = OCIAttrGet((dvoid *)srvhp,
   OCI_HTYPE_SERVER,
   (dvoid *)&serverStatus, (ub4 *)0, OCI_ATTR_SERVER_STATUS,
   errhp);if (serverStatus ==
   OCI_SERVER_NORMAL)printf("Connection is up.\n");else if (serverStatus ==
   OCI_SERVER_NOT_CONNECTED) printf("Connection is down.\n");
```

**Steps for Using Application Continuity**

Perform these steps to use Application Continuity:

- As a prerequisite, enable and configure Application Continuity or Transparent Application Continuity (TAC) for your database service on Autonomous Database. See Configure Application Continuity on Autonomous Database for more information.
- Oracle strongly recommends that you use the latest client drivers. Oracle Database 19c client drivers and later provide full support for Application Continuity (AC) and for Transparent Application Continuity (TAC). Use one of the following supported clients drivers:
  - Oracle JDBC Replay Driver 19c or later. This is a JDBC driver feature provided with Oracle Database 19c for Application Continuity
- Oracle Universal Connection Pool (UCP) 19c or later with Oracle JDBC Replay Driver 19c or later
- Oracle Weblogic Server 12c with Active GridLink, or third-party JDBC application servers using UCP with Oracle JDBC Replay Driver 19c or later
- Java connection pools or standalone Java applications using Oracle JDBC Replay Driver 19c or later
- Oracle Call Interface Session Pool 19c or later
- Oracle Call Interface based applications using 19c OCI driver or later
- SQL*Plus 19c (19.8) or later
- ODP.NET pooled, Unmanaged Driver 19c or later ("Pooling=true" default in 12.2 and later)
- Oracle Call Interface based applications using 19c OCI driver or later

**Return Connections to the Connection Pool**

The application should return the connection to the Oracle connection pool on each request. Best practice for application usage is to check-out (borrow) connections for only the time that they are needed, and then check-in to the pool when complete for the current actions. This is important for best application performance at runtime, for rebalancing work at runtime and during maintenance and failover events. This practice is also important for draining.

When using an Oracle connection pool, such as Universal Connection Pool (UCP) or OCI Session Pool, or ODP.Net Unmanaged Provider or when using WebLogic Active GridLink, following this practice embeds request boundaries that Application Continuity uses to identify safe places to resume and end capture. This is required for Application Continuity and is recommended for Transparent Application Continuity.

Transparent Application Continuity, in addition, will discover request boundaries if a pool is not in use or when replay is disabled. The conditions for discovering a boundary are:

- No open transaction
- Cursors are returned to the statement cache or cancelled
- No un-restorable session state exists (refer to Clean Session State between Requests in this paper)

**Enable Mutables Used in the Application**

Mutable functions are functions that can return a new value each time they are executed. Support for keeping the original results of mutable functions is provided for `SYSDATE`, `SYSTIMESTAMP`, `SYS_GUID`, and `sequence.NEXTVAL`. If the original values are not kept and different values are returned to the application at replay, replay is rejected.

If you need mutables for PL/SQL, issue `GRANT KEEP` as required.

For example:

```
SQL> GRANT KEEP DATE TIME to adb_user;
SQL> GRANT KEEP SYSGUID to adb_user;
SQL> GRANT KEEP SEQUENCE mySequence to adb_user on mysequence.myobject;
```

**Side Effects**

When a database request includes an external call such as sending MAIL or transferring a file then this is termed a side effect.
Side effects are external actions, they do not roll back. When replay occurs, there is a choice as to whether side effects should be replayed. Many applications choose to repeat side effects such as journal entries and sending mail as duplicate executions cause no problem. For Application Continuity side effects are replayed unless the request or user call is explicitly disabled for replay. Conversely, as Transparent Application Continuity is on by default, TAC does not replay side effects. The capture is disabled, and re-enables at the next implicit boundary created by TAC.

Developer Best Practices for Continuous Availability

Follow these best practices to code for continuous availability on Autonomous Database.

Return Connections to the Connection Pool

The most important developer practice is to return connections to the connection pool at the end of each request. This is important for best application performance at runtime, for draining work and for rebalancing work at runtime and during maintenance, and for handing failover events. Some applications have a false idea that holding onto connections improves performance. Holding a connection neither performs nor scales.

Clean Session State between Requests

It is best practice to clean session state between database requests.

When an application returns a connection to the connection pool, cursors in FETCH status, and session state set on that session remain in place unless an action is taken to clear them. If your application is setting state, it is best practice to return your cursors to the statement cache and to clear application related session state to prevent leakage to later re-uses of that database session. Cleaning your session state ensures that TAC can discover boundaries.

To automatically clean your state between requests with Oracle Database 21c, set the service attribute RESET_STATE=LEVEL1. Doing this will avoid state leakage and fetching from cursors by later usage of the connection pool.

If you are using Oracle Database 19c, use DBMS_SESSION.RESET_PACKAGE to clear PL/SQL global variables, use TRUNCATE to clear temporary tables, SYS_CONTEXT.CLEAR_CONTEXT to clear context and cancel your cursors by returning them to the statement cache.

If your application is stateless, such as REST, APEX, Microservice, and most web applications, it is best practice to use RESET_STATE.

Do not embed COMMIT in PL/SQL and Avoid Commit on Success and Autocommit

It is recommended practice to use a top-level commit, (O)COMMIT or COMMIT() or OCITransCommit). If your application is using COMMIT embedded in PL/SQL or AUTOCOMMIT or COMMIT ON SUCCESS, it may not be possible to recover following an outage or timeout. PL/SQL is not reentrant. Once a commit in PL/SQL has executed, that PL/SQL block cannot be resubmitted. Applications either need to unpick the commit which is not sound as that data may have been read, or for batch use a checkpoint and restart technique. When using AUTOCOMMIT or COMMIT ON SUCCESS, the output is lost.
If your application is using a top-level commit, then there is full support for Transparent Application Continuity (TAC), Application Continuity (AC), and TAF Select Plus. If your application is using COMMIT embedded in PLSQL or AUTOCOMMIT or COMMIT ON SUCCESS, it may not be possible to replay for cases where that the call including the COMMIT did not run to completion.

Use ORDER BY or GROUP BY in Queries

Application Continuity ensures that the application sees the same data at replay. If the same data cannot be restored, Application Continuity will not accept the replay. When a SELECT uses ORDER BY or GROUP BY order is preserved. In Autonomous Database the query optimizer most often uses the same access path, which can help in the same ordering of the results. Application Continuity also uses an AS OF clause to return the same query results where AS OF is allowed.

Considerations for SQL*Plus

SQL*Plus is often our go to tool for trying things out. SQL*Plus of course does not reflect our actual application that will be used in production, so it is always better to use the real application test suite to test your failover plan and to measure your protection. SQL*Plus is not a pooled application so does not have explicit request boundaries. Some applications do use SQL*Plus for example for reports. To use SQL*Plus with failover check the following:

1. FAN is always enabled for SQL*Plus. Use the recommended connect string that auto-configures ONS end points for you.

2. When using SQL*plus the key is to minimize round trips to the database: https://blogs.oracle.com/opal/sqlplus-12201-adds-new-performance-features

3. SQL*Plus is supported for TAC starting with Oracle Database 19c. For best results set a large arraysize. For example (set arraysize 1000). Avoid enabling serveroutput as this creates unrestorable session state.

Notes for Application Continuity on Autonomous Database

There are restrictions for Application Continuity on Autonomous Database, as follows:

- By default Application Continuity is disabled on Autonomous Database. See Configure Application Continuity on Autonomous Database for more information.

- For JDBC clients using Oracle Database JDBC driver and jars, draining for Java based applications on Autonomous Database requires that you use Oracle JDBC driver and Companion Jars 19.12.0.0 or newer, or apply a patch. See Patch Request 31112088 for more information.
Calling Web Services from Autonomous Database

Describes options for calling Web Services from Autonomous Database.

There are a number of options for calling Web Services from Autonomous Database, including the following:

- **Use DBMS_CLOUD REST APIs:** The `DBMS_CLOUD.SEND_REQUEST` function begins an HTTP request, gets the response, and ends the response. This function provides a workflow for sending a cloud REST API request with arguments and the function returns a response code and payload. See `SEND_REQUEST Function and Procedure` for more information.

- **Use Oracle APEX:** You can interact with both SOAP and RESTful style web services from APEX in your Autonomous Database instance. See `Use Web Services with Oracle APEX` for more information.

- **Use UTL_HTTP to submit a request to a public site:** See `Submit an HTTP Request to a Public Host with UTL_HTTP` for more information.

- **Use UTL_HTTP to submit a request to a private site:** See `Submit an HTTP Request to a Private Host with UTL_HTTP` for more information.

See `PL/SQL Packages` for information on restrictions for `UTL_HTTP` on Autonomous Database.

Submit an HTTP Request to a Public Host with UTL_HTTP

Provides examples to submit an HTTP request on a public host.

Submit an HTTP request for a public host `www.example.com`:

```sql
-- Create an Access Control List for the host
BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
        host => 'www.example.com',
        ace => xs$ace_type(privilege_list => xs$name_list('http'),
                        principal_name => 'ADMIN',
                        principal_type => xs_acl.ptype_db));
END;
/
-- Set Oracle Wallet location (no arguments needed)
BEGIN
    UTL_HTTP.SET_WALLET('');
END;
/
-- Submit an HTTP request
SELECT UTL_HTTP.REQUEST(url => 'https://www.example.com/') FROM dual;
```
Note:

If your Autonomous Database instance is on a private endpoint and you want your 
UTL_HTTP calls to public hosts to be subject to your private endpoint 
VCN's egress rules, set the ROUTE_OUTBOUND_CONNECTIONS database property 
to PRIVATE_ENDPOINT. See Enhanced Security for Outbound Connections 
with Private Endpoints for more information.

See PL/SQL Packages for information on restrictions for UTL_HTTP on Autonomous 
Database.

Submit an HTTP Request to a Private Host with UTL_HTTP

Provides examples to submit an HTTP request on a private host.

To submit a UTL_HTTP request to a target host on a private endpoint, the target host 
must be accessible from the source database's Oracle Cloud Infrastructure VCN. For 
example, you can connect to the target host when:

• Both the source database and the target host are in the same Oracle Cloud 
  Infrastructure VCN.
• The source database and the target host are in different Oracle Cloud 
  Infrastructure VCNs that are paired.
• The target host is an on-premises network that is connected to the source 
  database's Oracle Cloud Infrastructure VCN using FastConnect or VPN.

To make a UTL_HTTP call to a target on a private endpoint, make sure the following 
ingress and egress rules are defined:

• Define an egress rule in the source database's subnet security list or network 
  security group such that the traffic to the target host is allowed on port 443.
• Define an ingress rule in the target host's subnet security list or network security 
  group such that the traffic from the source database's IP address to port 443 is 
  allowed.
Making **UTL_HTTP** calls to a private host is only supported in commercial regions and US Government regions. This feature is enabled by default in all commercial regions.

This feature is enabled by default in US Government regions for newly provisioned databases.

For existing US Government databases on a private endpoint, if you want to make **UTL_HTTP** from an Autonomous Database to a target in a US Government region, please file a Service Request at Oracle Cloud Support and request to enable the private endpoint in government regions database linking feature.

US Government regions include the following:

- Oracle Cloud Infrastructure US Government Cloud with FedRAMP Authorization
- Oracle Cloud Infrastructure US Federal Cloud with DISA Impact Level 5 Authorization

To make a **UTL_HTTP** call to a target on a private endpoint, use **DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE** and specify the **private_target** parameter with value **TRUE**. For example:

```sql
-- Create an Access Control List for the host
BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
        host => 'www.example.com',
        ace => xs$ace_type(privilege_list => xs$name_list('http'),
            principal_name => 'ADMIN',
            principal_type => xs_acl.ptype_db),
        private_target => TRUE);
END;
/

-- Set Oracle Wallet location (no arguments needed)
BEGIN
    UTL_HTTP.SET_WALLET('');
END;
/

-- Submit an HTTP request
SELECT UTL_HTTP.REQUEST(url => 'https://www.example.com/',
    https_host => 'www.example.com') FROM dual;
```

If you set **ROUTE_OUTBOUND_CONNECTIONS** to **PRIVATE_ENDPOINT**, setting the **private_target** parameter to **TRUE** is not required in this API. See Enhanced Security for Outbound Connections with Private Endpoints for more information.
See PL/SQL Packages for information on restrictions for UTL_HTTP on Autonomous Database.
Using Oracle Java on Autonomous Database

Autonomous Database supports Oracle JVM. Oracle JVM is a standard, Java-compatible environment that runs any pure Java application.

Oracle JVM is compatible with the standard JLS and the JVM specifications. It supports the standard Java binary format and the standard Java APIs. In addition, Oracle Database adheres to standard Java language semantics, including dynamic class loading at run time.

See About Using Java in Oracle Database for information on Oracle Java.

Enable Oracle Java

Use `DBMS_CLOUD_ADMIN.ENABLE_FEATURE` to enable Oracle Java on Autonomous Database.

1. Run `DBMS_CLOUD_ADMIN.ENABLE_FEATURE`.

   ```sql
   BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_FEATURE(
      feature_name => 'JAVAVM' );
   END;
   /
   
   ```

   This initiates the request to install Oracle Java on the Autonomous Database instance.

   See `ENABLE_FEATURE Procedure` for more information.

2. Restart the Autonomous Database instance.

   See `Restart Autonomous Database` for more information.

   After you restart the Autonomous Database instance, Oracle JVM is enabled on the instance.

Check Oracle Java Version

You can check the Oracle Java version and the component registry for information on Oracle Java in the Autonomous Database instance.

1. Check the component registry status and version for Oracle Java.

   ```sql
   SELECT status, version FROM DBA_REGISTRY
   WHERE comp_id = 'JAVAVM';
   
   STATUS   VERSION
   ------   ----------
   VALID    19.0.0.0.0
   
   ```

   If Oracle Java is not installed, this query shows no rows.
2. Check the Oracle Java JDK version.

```sql
SELECT dbms_java.get_jdk_version FROM DUAL;
```

```
GET_JDK_VERSION
---------------
1.8.0_331
```

**Notes for Oracle Java on Autonomous Database**

Provides notes for using Oracle Java on Autonomous Database.

- You cannot disable Oracle Java after it is enabled on the Autonomous Database instance.
- Autonomous Database performs Oracle Java patching, as required, during the regular Autonomous Database maintenance window.

During Oracle Java patching, Java is not available and users could get an error similar to the following:

```
ERROR at line 1:
ORA-29548: Java system class reported: release of Java system classes in the database (19.0.0.0.220118 1.8) does not match that of the oracle executable (19.0.0.0.220419 1.8).
```

During the maintenance window, during the Java patching phase there would be no response for Java session calls or you would see an ORA-29548 error. After the maintenance window completes, Java usage is restored.

See [About Scheduled Maintenance and Patching](#) for more information.
Using Oracle Real Application Testing - Database Replay

Oracle Real Application Testing is an extremely cost-effective and easy-to-use proactive performance management solution that enables businesses to fully assess the outcome of a system change in test or production.

**Note:**
Only the Database Replay component of Real Application Testing is covered in this section.

**Topics**
- About Oracle Database Replay
- Capture a Workload on a non-Autonomous Database
- Replay a Workload on Autonomous Database

**About Oracle Database Replay**

You can use Database Replay to capture a workload on the production system and replay it on a test system with the exact timing, concurrency, and transaction characteristics of the original workload.

Database Replay enables you to test the effects of a system change on a workload without affecting the production system.

Database Replay captures workload on a production system and simulates the same workload on a test system.

This provides an accurate method to test the impact of a variety of system changes.

You can use Oracle Database Replay to capture workload from an on-premises, or other cloud service database and replay it on an Autonomous Database. This enables you to compare workloads between an on-premises database (or other cloud service database) and Autonomous Database.

**Capture a Workload on a non-Autonomous Database**

The first step in using Database Replay is to capture the production workload.

When you begin workload capture, all requests from external clients directed to Oracle Database are tracked and stored in binary files called capture files.

A workload capture results in the creation of two subdirectories, `cap` and `capfiles`, which contains the capture files.
The capture files provide all pertinent information about the client request, including transaction details, bind values, and SQL text.

These capture files are platform independent and can be transported to another system.

See Capturing a Database Workload for more information on workload capture.

Replay a Workload on Autonomous Database

After you complete a workload capture, you can replay it on a test system. During workload replay, Oracle replays the actions recorded during workload capture on the test system with the same timing, concurrency, and transaction dependencies of the production system.

Run `DBMS_CLOUD_ADMIN.REPLAY_WORKLOAD` to initiate a workload replay captured from an on-premises or other cloud service database on your Autonomous Database.

Example to replay on Autonomous Database a workload captured from an on-premises or other cloud service database:

```sql
BEGIN
  DBMS_CLOUD_ADMIN.REPLAY_WORKLOAD(
    location_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
    credential_name => 'CRED_TEST',
    synchronization => TRUE,
    process_capture => TRUE);
END;
/
```

This downloads the capture files contained in the Object Storage location specified in the `location_uri` parameter, and replays the workload capture from the capture files. The replay generates and uploads the replay and Automatic Workload Repository reports to the Object Storage location specified in the `location_uri` parameter.

The `credential_name` parameter specifies the credential to access the object storage bucket. The credential that you supply must have the write privileges to write into the Object Storage bucket to upload the replay and the Automatic Workload Repository reports to the bucket.

If you do not specify a `credential_name` value, then `DEFAULT_CREDENTIAL` is used.

The `synchronization` parameter specifies the synchronization method used during workload replay. A `TRUE` value indicates that the synchronization is SCN based.

The `process_capture` specifies whether you need to specify `process_capture` value or not. A `TRUE` value indicates that the replay includes `process_capture`.

Note:

You must maintain the same logical state of the capture and replay databases at the start of the capture time.
In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

See Navigate to Oracle Cloud Infrastructure Object Storage and Create Bucket for more information on Object Storage.

See Upload Files to Your Oracle Cloud Infrastructure Object Store Bucket for more information on uploading files to Object Storage.

The credential_name you use in this step is the credentials for the Object Store.

You don't need to create a credential to access Oracle Cloud Infrastructure Object Store if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

See REPLAY_WORKLOAD Procedure for more information.
Using and Managing a Cloud Code Repository with Autonomous Database

Autonomous Database provides routines to manage and store files in Cloud Code (Git) Repositories. The supported Cloud Code Repositories are: GitHub, AWS CodeCommit, and Azure Repos.

Topics

• About Cloud Code Repositories with Autonomous Database
• Initialize a Cloud Code Repository
• Create and Manage a Cloud Code Repository
• Use File Operations with a Cloud Code Repository
• Use SQL Install Operations with a Cloud Code Repository

About Cloud Code Repositories with Autonomous Database

The `DBMS_CLOUD_REPO` package provides a single interface for accessing a Cloud Code Repository from Autonomous Database.

The supported Cloud Code Repositories provide the following features:

• Git Version Control System: Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

• Git Repository: A Git repository is a virtual storage of your project. It allows you to save versions of your code, which you can access when needed.

The `DBMS_CLOUD_REPO` APIs use a repository handle (`REPO` object). The repository handle is an opaque JSON object that represents a Cloud Code Repository of a specific cloud provider. A `REPO` object can be passed to different `DBMS_CLOUD_REPO` APIs. This opaque object ensures that `DBMS_CLOUD_REPO` procedures and functions are multicloud compatible; you do not have to change your code when you migrate from one Cloud Code Repository provider to another Cloud Code Repository.

Autonomous Database provides the following to help you work with Cloud Code Repositories:

• Repository initialization operations to allow you to initialize a repository. See Initialize a Cloud Code Repository for more information.

• Repository management operations that let you create, list, update or delete a repository. See Create and Manage a Cloud Code Repository for more information.

• Repository file management operations to upload, download, update, and delete files. See Use File Operations with a Cloud Code Repository for more information.
• SQL install operations that let you export database object metadata DDL to a repository and install SQL statements into the database from a Cloud Code Repository.

See Use SQL Install Operations with a Cloud Code Repository for more information.

Initialize a Cloud Code Repository

The DBMS_CLOUD_REPO initialization routines initialize a Cloud Code Repository. After you obtain a Cloud Code Repository handle, you use the handle to access the Cloud Code Repository.

To initialize a Cloud Code Repository:

1. Create a credential to access the Cloud Code Repository.

   See CREATE_CREDENTIAL Procedure for information on creating credentials.

2. Depending on the repository, GitHub, Azure Repos, or AWS CodeCommit, call DBMS_CLOUD_REPO.INIT_REPO with the parameters for the particular repository to obtain a repository handle.

The following examples provide samples for each supported Cloud Code Repository.

• GitHub Initialization:

```sql
DEFINE repo_name='test_repo';
DEFINE cred_name='GITHUB_CRED';
VAR repo clob
BEGIN
 :repo := DBMS_CLOUD_REPO.INIT_REPO(
   params => JSON_OBJECT('provider' value 'github',
     'repo_name' value
     '&repo_name',
     'credential_name' value
     '&cred_name',
     'owner' value
     '<myuser>')
 );
END;
/
```

• Azure Repos Initialization:

```sql
DEFINE repo_name='test_repo';
DEFINE cred_name='AZURE_REPO_CRED';
VAR repo clob
BEGIN
 :repo := DBMS_CLOUD_REPO.INIT_REPO(
   params => JSON_OBJECT('provider' value 'azure',
     'repo_name' value
     '&repo_name',
     'credential_name' value
     '&cred_name',
     'organization' value
     '<myuser>')
 );
END;
/```
• **AWS CodeCommit Initialization:**

```sql
DEFINE repo_name='test_repo';
DEFINE cred_name='AWS_REPO_CRED';
VAR repo clob
BEGIN
    :repo := DBMS_CLOUD_REPO.INIT_REPO(
        params => JSON_OBJECT('provider' value 'aws',
                                'repo_name' value '&repo_name',
                                'credential_name' value
                                '&cred_name',
                                'region' value 'us-east-1')
    );
END;
/
```

See [DBMS_CLOUD_REPO Initialization Operations](#) for details on the initialization functions.

---

## Create and Manage a Cloud Code Repository

The [DBMS_CLOUD_REPO](#) management routines allow you to manage a Cloud Code Repository by creating, listing, updating, or deleting a repository.

First, obtain a Cloud Code Repository handle to provide access a repository. See [Initialize a Cloud Code Repository](#) for details.

1. To create a repository:

   ```sql
   VAR repo clob
   BEGIN
     DBMS_CLOUD_REPO.CREATE_REPOSITORY(
         repo => :repo,
         description => 'test repo'
     );
   END;
   /
   ```

2. To update a repository:

   ```sql
   VAR repo clob
   DEFINE repo_name='test_repo';
   BEGIN
     DBMS_CLOUD_REPO.UPDATE_REPOSITORY(
         repo => :repo,
         new_name => '&repo_name' || '_new'
     );
   END;
   ```
3. To list repositories:

```sql
col id format a30
col name format a10
col description format a15
select id, name, bytes, private, description from
    DBMS_CLOUD_REPO.LIST_REPOSITORIES(:repo);
```

4. To delete a repository:

```sql
VAR repo clob
BEGIN
    DBMS_CLOUD_REPO.DELETE_REPOSITORY(
        repo => :repo
    );
END;
/
```

See DBMS_CLOUD_REPO Repository Management Operations for more information.

Use File Operations with a Cloud Code Repository

The DBMS_CLOUD_REPO file operations allow you to create, get, list, update, or delete files in a Cloud Code Repository.


You also need to create a repository before you work with files. See Create and Manage a Cloud Code Repository for details.

1. To get a file:

```sql
SELECT DBMS_CLOUD_REPO.GET_FILE(repo => :repo, file_path => 'test1.sql')
```

2. To create a file:

```sql
VAR repo clob
BEGIN
    DBMS_CLOUD_REPO.PUT_FILE(
        repo => :repo,
        file_path => 'test1.sql',
        contents => UTL_RAW.cast_to_raw('create table t1 (x varchar2(30))' || CHR(10) || ')
    );
END;
/
```
3. To update a file:

```sql
VAR repo clob
BEGIN
    DBMS_CLOUD_REPO.PUT_FILE(
        repo => :repo,
        file_path => 'test1.sql',
        contents => UTL_RAW.cast_to_raw('create table t2 (x varychar2(30))' || CHR(10) || '/'),
    );
END;
/
```

4. To list files:

```sql
SELECT id, name, bytes, url FROM DBMS_CLOUD_REPO.LIST_FILES(repo => :repo);
```

5. To delete a file:

```sql
VAR repo clob
BEGIN
    DBMS_CLOUD_REPO.DELETE_FILE(
        repo => :repo,
        file_path => 'test1.sql',
    );
END;
/
```

See DBMS_CLOUD_REPO File Operations for more information.

Use SQL Install Operations with a Cloud Code Repository

The DBMS_CLOUD_REPO SQL Install operations allow you to store and download SQL scripts from a Cloud Code Repository.

Obtain a Cloud Code Repository handle before using the SQL Install operations. See Initialize a Cloud Code Repository for details.

You also need to create a repository before you work with SQL Install operations. See Create and Manage a Cloud Code Repository for details.

The scripts are intended as schema install scripts and not as generic SQL scripts:

- Scripts cannot contain SQL*Plus client specific commands.
- Scripts cannot contain bind variables or parameterized scripts.
- SQL statements must be terminated with a slash on a new line (/).
- Scripts can contain DDL, DML PLSQL statements, but direct SELECT statements are not supported. Using SELECT within a PL/SQL block is supported.

Any SQL statement that can be run using EXECUTE IMMEDIATE will work if it does not contain bind variables or defines.
1. To upload DDL metadata to a Cloud Code Repository:

```sql
VAR repo clob
BEGIN
    DBMS_CLOUD_REPO.EXPORT_OBJECT(
        repo => :repo,
        object_type => 'PACKAGE',
        object_name => 'MYPACK',
        file_path   => 'mypack.sql'
    );
END;
/
```

2. To install SQL statements from a file:

```sql
VAR repo clob
BEGIN
    DBMS_CLOUD_REPO.INSTALL_FILE(
        repo => :repo,
        file_path     => 'test3.sql',
        stop_on_error => FALSE
    );
END;
/
```

3. To install SQL statements from a buffer:

```sql
VAR repo clob
BEGIN
    DBMS_CLOUD_REPO.INSTALL_SQL(
        repo => :repo,
        content   => 'create table t1 (x varchar2(30))' || CHR(10) || '/*',
        stop_on_error => FALSE
    );
END;
/
```

See DBMS_CLOUD_REPO SQL Install Operations for more information.
Part II

Load, Explore, and Analyze Data with Data Tools on Autonomous Database

This part provides information on using the data tools provided with Database Actions to load, explore, analyze, and model data.

See Connect with Built-in Oracle Database Actions for information on accessing Oracle Database Actions from Autonomous Database.

Topics

- The Data Load Page
- The Catalog Page
- The Data Insights Page
- The Data Analysis Tool
Use the Data Load page to make more data available to your Oracle Autonomous Database. You can load data from files or databases, from links to external databases or cloud storage files, or from a live feed of data from cloud storage.

From the Data Load page, you can also explore the data in your autonomous database and manage your cloud storage locations.

To reach the Data Load page, click **Data Load** in the Database Actions page, or click the **Selector** icon and select **Data Load** from the Data Tools menu in the navigation pane.

To load or create links to data or create live table feeds, on the Data Load page, select a combination of an operation and a data source location. The following table lists the operations and the source locations that support those operations.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Source Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load data</td>
<td>Local file</td>
<td>Load data from files on your local device, from remote databases, or from cloud storage into tables in your Oracle Autonomous Database.</td>
</tr>
<tr>
<td></td>
<td>Database</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cloud storage</td>
<td></td>
</tr>
<tr>
<td>Link data</td>
<td>Database</td>
<td>Create external tables or views in your Oracle Autonomous Database that link to data in cloud storage or remote databases. Changes to the source data automatically appear in the target objects.</td>
</tr>
<tr>
<td></td>
<td>Cloud storage</td>
<td></td>
</tr>
<tr>
<td>Feed data</td>
<td>Cloud storage</td>
<td>Set up a feed of data from a cloud storage bucket into a table. Changes to the source data load into the target table as scheduled or on demand.</td>
</tr>
</tbody>
</table>

The following topics describe these actions.

- Exploring Data
- Managing Cloud Storage Connections
- Loading Data
- Linking Data
- Feeding Data

**Checking Data Load Jobs**

You can check an existing data load job and retrieve the data load job later when required.
The Data Load Job page displays the data load history in your Oracle Autonomous Database. On the Data Load Page, in the Explore and Connect section, select Data Load Jobs.

The Data Load Job page contains the Search for Data Load Jobs field, a list of Data load job cards. You can enter the data load job you are looking for in the field or click one of the data load jobs from the list.

The Data Load Jobs page consists of:

1. **Search for Data Load Jobs field**
   
   You can click the field and type or search for the name of the data load job you are looking for.

2. **Toolbar**
   
   The toolbar consists of the following buttons:
   
   - **Sort By**
     
     To select sorting values, click the Sort By button to open the list of options. Then click the Ascending or Descending icon next to one or more of the sorting values. For example, if you select the Ascending icon next to Entity name and the Descending icon next to Entity type, the entities will be sorted in alphabetical order by entity name and then in reverse alphabetical order by entity type.
     
     Click Reset in the list to clear the choices in the list.
     
     The sorting values you choose are listed next to the Sort by label beneath the toolbar. Click the X icon on a sorting value to remove it.
   
   - **Page size**
     
     By default, up to 25 entities are displayed on the page. If you want more entities on a page, select a number from this list.
   
   - **Previous and Next**
     
     If the search results are displayed on multiple pages, click these buttons to navigate through the pages.
   
   - **Refresh**
Click to refresh the data load jobs shown on the page, based on the current search field.

3. **Filters** panel

Select one or more filter values to limit the data load jobs shown on the page. Only those entities that match the filter values are shown. The filter criteria are based on the schemas and how data is loaded or linked. That is, the items returned by a search are filtered by these filter settings. Selecting all or none of the options shows all entities.

4. **Display Area**

The area beneath the **Search for Data Load Jobs** field displays the data load job carts returned by a search and that match the filter criteria set in the **Filters** panel. It displays a list of carts which represents a list of previously run data load jobs.

You can view details about the job, re-run the Data Load job, Rename the Data Load Job and Delete the Data Load Job.

**View Details** about the Data Load Job

To view details about the existing data load job, click **Action** and select **View Details** in the card for the load job. Selecting **View Details** displays details like **Lineage**, **Impact** and **Log details** of the data load job.

For details on Lineage, Impact and Log details, see **Viewing Entity Details**.

**Rerun Data Load Job**

After viewing details about the selected data load or data link job whose sources are from cloud storage, you can re-run the selected previously run data load job. The previous files and folders processed in that job will be loaded in the cart with all your settings unchanged from the last run. On the left side of the page is a navigator pane, where you choose the cloud store connection and the folders or files containing the data that were previously run. The previously used cloud storage details are already present in the navigator pane. Select additional files or folders from the navigator pane and drop them in the Data Load Cart area.

To change the cloud storage connection and the folders and files containing the data, see **Managing Cloud Storage Connections**.

Once you have added the data sources to the data load cart, click the **Start** icon in the data load cart menu bar. When the data load job completes, the Load Cloud Object page displays the results of the job. At the top of the page, a status message shows the number of items for which the load has completed over the number of items in the job and the total time elapsed for the job. If any previously loaded or linked files and folders are no longer present in the cloud location, an error message reports the problem. You can view the list of unavailable files and folders, which will not be loading into the cart for processing.

Click the **Cancel** icon to cancel re-running the selected job.

**Note:**

The **Rerun Data Load Job** option is visible for data jobs with Load and Link data options for Object storage.

**Rename Data Load Job**

To rename existing name of data load job cart, select the **Rename Data Load Job** option. The data load job names are system generated. This option enables you to rename the job to
be more descriptive. This makes it easier to search the next time you want to re-run
the data load job.

**Delete Data Load Job**

Select the **Delete Data Load Job** option to remove the previously run job from the
Data load Job page. This will also remove the log files and the error logs generated
when attempting to run the data load job.

**Exploring Data**

To view the data in the tables and views in your Oracle Autonomous Database, on the
Data Load page, in the Explore and Connect section, select **Explore**.

The navigator on the Explore page lists the tables and views in your Oracle
Autonomous Database. To filter the lists of tables and views displayed, enter a value in
the search field and press Enter. To refresh the list of tables and views, click the
Refresh icon.

To view details about a table or view, select the table or view in the navigator. The
Source Preview pane displays the columns and data of the table or view. For a table,
the pane includes a Statistics tab that displays the size and the number of rows and
columns of the table, the data type of the columns, the number of distinct values, and
other information. Below those details is a bar graph that displays the top unique
values for the selected column.

To close the Settings pane, click **Close**.

To return to the Data Load page, click **Data Load** at the top of the page.

**Managing Cloud Storage Connections**

A cloud storage link is a connection to a bucket in a cloud store. To view the existing
cloud storage links and to add new ones, on the Data Load page, select **CLOUD
LOCATIONS** and click **Next**.

The Manage Cloud page displays the existing cloud storage links, which are
CLOUD_STORAGE_LINK entities. You can edit or delete an link. You can also create
new cloud storage links.

You can sort the display of links. The links are also sorted by entity name, as indicated
by the **Sort by** list.

In the Sort by section, you can choose to sort the links by name, creation date, or both.

To remove a sorting value, deselect the sorting value on the Manage Cloud page.

To search for available cloud storage links, enter a value in the search field and press
Enter. The display then includes only the entities whose names contain the characters
in the search field. To clear the search field, click the Clear search results (X) icon in
the search field.

To refresh the display of cloud storage links, click the Refresh icon.

In the Add Cloud Storage pane, fields with an asterisk (*) are mandatory. To cancel the
creation of a link, click **Cancel**.
The procedure for creating a credential varies depending on the cloud storage provider. If your source files reside in a cloud store provided by one of the following, see the example for that provider.

- **Oracle Cloud Infrastructure (OCI)**, see Create an OCI Cloud Storage Link.
- **Amazon S3**, or you are calling an AWS API, see Create an Amazon S3 Cloud Storage Link.
- **Microsoft Azure Blob Storage** or you are calling an Azure API, see Create a Microsoft Azure Cloud Storage Link.
- **Google Cloud Storage**, see Create a Google Cloud Storage Link.
- **Other (Swift compatible) cloud storage**, see Create an Other (Swift Compatible) Cloud Storage Link.

Create an OCI Cloud Storage Link

1. On the Manage Cloud page, click **Add Cloud Storage**.
2. In the Storage Settings tab of the Add Cloud Store Location box, enter a name for the cloud storage link. For example:
   
   **My_Cloud_Store**

3. (Optional) In the **Description** field, enter a description for the link. For example:
   
   **My cloud storage link.**

4. From the **Cloud Store** drop-down list, select **Oracle**.
5. In the **URI + Bucket** field, enter the URI and bucket for your OCI instance bucket.
   a. To get the URI and bucket, go to the bucket in the Object Storage compartment in your Oracle Cloud Instance.
   b. In the Objects group, click the Actions (three vertical dots) icon for a file in the bucket, then click **View Object Details**.
   c. Copy all of the URL Path (URI) except for the file name. Be sure to include the trailing slash. For example, for the file `https://objectstorage.us-phoenix-1.oraclecloud.com/n/myoci/b/my_bucket/o/MyFile.csv`, select the following:

   ```
   https://objectstorage.us-phoenix-1.oraclecloud.com/n/myoci/b/my_bucket/o/
   ```

   d. Paste the string into the **URI + Bucket** field.
6. Select a credential option.
   - If you select **No Credential**, then proceed to the next step.
   - If you select **Select Credential**, then select a credential from the drop-down list.
   - If you select **Create Credential**, then do the following:
a. Enter a name in the **Credential Name** field. The name must conform to Oracle object naming conventions, which do not allow spaces or hyphens. For example:

```
my_credential
```

b. For an OCI cloud store, in the **Oracle Cloud Infrastructure User Name** field, enter your OCI user name. For example:

```
myUsername
```

c. For an OCI cloud store, in the **Auth Token** field, enter your auth token. For example:

```
LPB>Ktk(lM1SD+a]+r
```

For information on getting an auth token, see Working with Auth Tokens in Managing User Credentials.

7. Click **Next**. The dialog box progresses to the Cloud Data tab. This tab lists the objects available on this cloud storage location in the display area.

    **Note:**
    
The display area is blank when we create a new cloud storage location.

8. Click **Create** to create the cloud storage location.

**Create an Amazon S3 Cloud Storage Link**

1. On the Manage Cloud page, click **Add Cloud Storage**.

2. In the Storage Settings tab of the Add Cloud Store dialog box, enter a name for the cloud storage link.

```
My_Cloud_Store
```

3. (Optional) In the **Description** field, enter a description for the link. For example:

```
My cloud storage link.
```

4. From the **Cloud Store** drop-down list, select **Amazon S3**.

5. In the **URI + Bucket** field, enter the URI and bucket for your Amazon S3 bucket. For example:

```
https://s3-us-west-2.amazonaws.com/adwc/my_bucket
```

6. Select a credential option.
   
   - If you select **No Credential**, then proceed to the next step.
   
   - If you select **Select Credential**, then select a credential from the drop-down list.
If you select Create Credential, then do the following:

a. Enter a name in the Credential Name field. The name must conform to Oracle object naming conventions, which do not allow spaces or hyphens. For example:

```
my_credential
```

b. In the AWS Access Key ID field, enter your AWS access key ID. For example:

```
myAccessKeyID
```

c. In the AWS Secret Access Key field, enter your AWS secret access key. For information on AWS access keys, see Managing access keys for IAM users.

7. Click Next.
The dialog box progresses to the Cloud Data tab. This tab lists the objects available on this cloud storage location in the display area.

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The display area is blank when we create a new cloud storage location.</td>
</tr>
</tbody>
</table>

8. Click Create create the cloud storage location.

Create an Microsoft Azure Cloud Storage Link

1. On the Manage Cloud page, click Add Cloud Storage.

2. In the Storage Settings tab of the Add Cloud Store dialog box, enter a name for the cloud storage link. For example:

```
My_Cloud_Store
```

3. (Optional) In the Description field, enter a description for the link. For example:

```
My cloud storage link.
```

4. From the Cloud Store drop-down list, select Microsoft Azure.

5. In the URI + Bucket field, enter the URI and bucket for your Microsoft Azure bucket. For example:

```
https://objectstore.microsoft.com/my_bucket
```

6. Select a credential option.

   • If you select No Credential, then proceed to the next step.
   • If you select Select Credential, then select a credential from the drop-down list.
   • If you select Create Credential, then do the following:

     a. Enter a name in the Credential Name field. The name must conform to Oracle object naming conventions, which do not allow spaces or hyphens. For example:

     ```
     my_credential
     ```
b. In the **Azure Storage Account Name** field, enter the name of your Azure storage account. For example:

```
AZURE_KEY123...
```

c. In the **Azure Storage Account Access Key** field, enter your Azure access key.

For information on Azure storage accounts, see Create a storage account.

7. Click **Next**.

The dialog box progresses to the Cloud Data tab. This tab lists the objects available on this cloud storage location in the display area.

---

**Note:**

The display area is blank when we create a new cloud storage location.

8. Click **Create** to create the cloud storage location.

### Create a Google Cloud Storage Link

1. On the Manage Cloud page, click **Add Cloud Storage**.

2. In the Storage Settings tab of the Add Cloud Store dialog box, enter a name for the cloud storage link. For example:

```
My_Cloud_Store
```

3. (Optional) In the **Description** field, enter a description for the link. For example:

```
My cloud storage link.
```

4. From the **Cloud Store** drop-down list, select **Google**.

5. In the **URI + Bucket** field, enter the bucket and URI for your Google bucket. For example:

```
https://my_bucket.storage.googleapis.com
```

6. Select a credential option.

   - If you select **No Credential**, then proceed to the next step.
   - If you select **Select Credential**, then select a credential from the drop-down list.
   - If you select **Create Credential**, then do the following:
     a. Enter a name in the **Credential Name** field. The name must conform to Oracle object naming conventions, which do not allow spaces or hyphens. For example:

```
my_credential
```
b. In the **HMAC Access Key** field, enter your HMAC access ID. For example:

   GOOGTS1C3LPB3KTSDKMB2BFD

c. In the **HMAC Access Secret** field, enter your HMAC secret. For information on HMAC keys, see [HMAC Keys](#).

7. Click **Next**.
   The dialog box progresses to the Cloud Data tab. This tab lists the objects available on this cloud storage location in the display area.

   **Note:**
   The display area is blank when we create a new cloud storage location.

8. Click **Create** to create the cloud storage location.

**Create an Other (Swift Compatible) Cloud Storage Link**

1. On the Manage Cloud page, click **Add Cloud Storage**.

2. In the Storage Settings tab of the Add Cloud Store dialog box, enter a name for the cloud storage link. For example:

   My_Cloud_Store

3. (Optional) In the **Description** field, enter a description for the link. For example:

   My cloud storage link.

4. From the **Cloud Store** drop-down list, select **Other (Swift Compatible)**.

5. In the **URI + Bucket** field, enter the URI and bucket for your cloud store bucket. For example:

   https://someswiftcompatibleprovider.com/my_bucket

6. Select a credential option.
   • If you select **No Credential**, then proceed to the next step.
   • If you select **Select Credential**, then select a credential from the drop-down list.
   • If you select **Create Credential**, then do the following:
     a. Enter a name in the **Credential Name** field. The name must conform to Oracle object naming conventions, which do not allow spaces or hyphens. For example:

        my_credential

     b. In the **Access User Name** field, enter your access user name. For example:

        OTHER_KEY123...

     c. In the **Access Key** field, enter your access key.

7. Click **Next**.
The dialog box progresses to the Cloud Data tab. This tab lists the objects available on this cloud storage location in the display area.

**Note:**
The display area is blank when we create a new cloud storage location.

8. Click **Create** to create the cloud storage location.

---

### Loading Data with Autonomous Database

Describes packages and tools to load data with Autonomous Database.

**Topics**

- About Data Loading
- Load Data with Oracle Database Actions
- Load Data from Files in the Cloud
- Import Data Using Oracle Data Pump on Autonomous Database
- Load Data from Local Files with Oracle Database Actions
- Use Oracle GoldenGate to Replicate Data to Autonomous Database
- Load Data from Local Files Using SQL*Loader

---

### Loading Data From Local Files

To load data from local files into your Oracle Autonomous Database, on the Data Load page, select **LOAD DATA** and **LOCAL FILE**, then click **Next**. Drag one or more files from your local file system navigator and drop them in the Data Load Cart. You can also click **Select Files** or the Select Files icon, select one or more files from the file system navigator, and then click **Open**.

You can add files in these file formats: AVRO, CSV, JSON, TSV, delimited TXT, XLS, XLSX, XML. For information on supported file formats, see Format Specifications for JSON, AVRO, and XML Files.

An item for each file appears in the cart. For an XLS or XLSX spreadsheet, the worksheets of the spreadsheet appear as individual items. The item shows the name of the source file or worksheet and its size, and the name of the table that is the target for the data load.

You can add more files to the cart by clicking the Select Files icon. You can add any number of files to the cart and load data from all of them in a single data load job.

To remove a source file from the Data Load Cart, click the Remove (trash can) icon for the source item. To remove all source files from the cart, click the Remove All (trash can) icon in the Data Load Cart menu bar.

To return to the Data Load page, click **Data Load** above the Data Load Cart menu bar.
Specify Processing Options

To specify settings for the data load job or preview the data in the source or the target, click the Settings (pencil) icon for the item in the Data Load Cart.

In the settings pane, on the Settings tab, you can view the name and size of the file in the title of the Load Data dialog box.

The Name field specifies the name of the target table. The value in the field varies depending on the selection in the Options field. If the option is Create Table, then the default target value is the name of source file or worksheet. To specify a different name for the target table, enter it in the Name field. For the other target table choices in the Options field, the default value is <None>. Expand the drop-down list and select an existing table as the target.

In the Options field select Create Table, Insert into Table, Replace Data, Drop Table and Create New Table, or Merge into Table. Point to the question mark icon to see a brief description of the selected option.

The Source column name option specifies whether to get the source and target column names from the file or to specify the column names manually. Getting the column names from the header of the source file is the default. If you select the Get from file header option, then the first row in the file is processed as column names. If you deselect the option, then the first row is processed as data. To specify column names manually, enter a name for each target column in the Mapping section. You can also select a data type for the column.

The Rows to skip field specifies how many rows to skip when loading the source data into the target. If you have selected the Get from file header option, and if you enter a number greater than 0 in the Rows to skip field, then that number of rows after the first row are not loaded into the target. If you have deselected the Get from file header option, and if you enter a number greater than 0 in the Rows to skip field, that number of rows including the first row are not loaded into the target.

To change the character set encoding for the contents of the file, select a value from the Encoding drop-down list.

To specify the characters that enclose text, select the double-quotes or single-quote character or None from the Text enclosure drop-down list.

To change the delimiter character that separates columns in the source, expand the Field delimiter drop-down list and select a character. For example, if the file has columns delimited by semicolons, change the delimiter from the default comma delimiter to a semicolon.

To convert any invalid value in a numeric source column to a null value in the target column, select the Numeric column Convert invalid data to null option.

Specify Mappings

If you select the Create Table or Drop Table and Create New Table option and you are getting the source column names from the file header, then in the Mapping section either accept the default values for the target columns and data types or specify different values. To specify different values, in the target column, enter a name for the column. In the Data Type column, select a data type from the drop-down list. If you are not getting the source column names from the file header, then for each source column specify a name for the target column and select a data type from the Data Type drop-down list. For the Date data type, select a date format from the Format drop-down list.

For the Merge into Table option, for each source column, select a target column form the drop-down list. You must specify at least one column as a key column. To specify a column
as a key column, select the Merge Key check box for the column. Merge keys are one or more columns that uniquely identify each row in the table. Merge keys must not contain any null values. For loading tables with primary keys, this option automatically enables the selection of primary key columns as the merge keys.

For the Insert into Table or Replace Data options, for each source column, select a target column from the drop-down list of existing columns.

**Preview Source Data**

To view a selection of data in the source file, select the File tab. The source preview has a Preview size field and a portion of the data in the file. To change the number of rows displayed, you can enter a value from 1 to 100 in the field.

Any modifications you make in the source preview do not affect the loading of data from the file.

**Preview Target Data**

For all options except Create Table, to view the existing data in the target table, select the Target Preview tab. The target preview displays the data in the target table before you run the data load job.

To close the settings pane, click Close.

**Run the Data Load Job**

When you have added all of the sources for the job and specified the settings for each source, to run the job click the Start icon in the Data Load Cart menu bar. In the Run Data Load Job dialog box, click Run. To stop the data load job, click the Stop icon.

When the data load job completes, the Local Files page displays the results of the job. At the top of the page, the Status shows the number of items for which the load has completed over the number of items in the job, and the total time elapsed for the job.

To view information about an item in the job, click the Settings icon in the item. In the settings pane, the Settings tabs are the same as before running the job, except that the target preview now contains the data loaded by the data load job and a Data Definition tab. To close the pane, click Close.

To view a log of the load operation, click the Logging icon. You can save the log, clear it, or refresh it. Click OK to dismiss the log.

The list of tables on the Data Load / Explore page contains any new tables created. The target tables for the Insert into Table, Replace Data, Drop Table and Create New Table, and Merge into Table options contain the loaded data.

Click the Explore Catalog button on the Local files page to view the new or updated table on the Catalog page.

**Fixing a data load job**

After your data load job, you might see errors that you want to correct, or upon inspection, realize that you wanted to name a column differently. In such cases, click the Reload Cart option to reload cards from your recent cart and edit them as you did before your first attempt. Click the pencil icon to make any changes to the data load job (i.e., change a column name).

Click Done to return to the Database Actions page.
Loading Data from Other Databases

To load data from tables in another database into your Oracle Autonomous Database, on the Data Load page, select **LOAD DATA** and **DATABASE**, then click **Next**. Select a database link from the drop-down list. Drag one or more tables from the list of database tables and drop them in the Data Load Cart.

Each table appears as an item in the Data Load Cart. The item shows the name of the table and the number of rows in it, and the name of the table that is the target for the data load.

To remove a table from the Data Load Cart, click the Remove (trash can) icon for the item. To remove all tables from the cart, click the Remove All (trash can) icon in the Data Load Cart menu bar.

To add a remote database to the list of database links, create a database link to the remote database. For information on creating a database link, see Database Links in *Oracle® Database Database Administrator's Guide 21c*.

The databases available to you appear in the drop-down list of the database navigation pane of the Load Tables page.

You can filter the tables displayed in the navigation pane by entering a case-sensitive value in the search field at the top of the navigation tree and pressing Enter. To display all of the tables again, clear the search field and press Enter.

You can add any number of tables from the navigation pane to the Data Load Cart and load data from all of them in a single data loading job. You can set filters on the data for a table to load only the specified data.

**Specify Processing Options**

To specify settings for the data load job, preview the data in the source or the target, and see statistics about the data, click the Settings (pencil) icon for the item in the Data Load Cart.

In the settings pane, on the Settings tab, you can view the name and size of the file in the title of the Load Data dialog box.

The **Table** field specifies the name of the target table. The value in the field varies depending on the selection in the **Options** field. If the option is **Create Table**, then the default target value is the name of source table. To specify a different name for the target, enter it in the **Name** field. For the other options, the default value is <None>. Expand the drop-down list and select a table as the target.

In the **Options** field for the source, select **Create Table**, **Insert into Table**, **Replace Data**, **Drop Table and Create New Table**, or **Merge into Table**. Point to the question mark icon to see a brief description of the selected option.

If you select **Create Table**, then in the **Name** field accept the default name, which is the name of the source table, or enter a different name.

If you select one of the other options, then expand the drop-down list of the **Name** field and select a table as the target.

You can set filters to load only the specified data from the source table. To set an initial filter condition on a table, do the following:

1. Click the Edit Source Filter icon.

2. In the Edit Source Filter dialog box, click **Add Filter Condition**.
3. In the Add Filter Condition dialog box, select a source column and an operator and specify a value.

4. Click Save.

To add another filter condition, click the Edit Source Filter icon and repeat the steps for adding a filter. The filter then has both filter conditions and the AND operator. You can change the AND to an OR by clicking the Edit Source Filter icon and manually replacing the AND with OR in the Filter field.

To remove a filter, in the Edit Source Filter dialog box, delete the value in the Filter field of the Edit Source Filter dialog box.

Specify Mappings

If you select the Create Table or the Drop Table and Create New Table option, then in the Mapping section either accept the default values for the target columns or specify different values. For the target column, enter a name for the column.

For the Insert into Table or Replace Data options, select a target column from the drop-down list of existing columns.

For the Merge into Table option, for each source column, select a target column from the drop-down list. You must specify at least one column as a key column. To specify a column as a key column, select the Merge Key check box for the column. Merge keys are one or more columns that uniquely identify each row in the table. Merge keys must not contain any null values. For loading tables with primary keys, this option automatically enables the selection of primary key columns as the merge keys.

Preview Source Data

To view the data in the source table, in the settings pane select the Source Table tab. The source preview displays the data in the table.

View Statistics

To view statistics about the source table, in the settings pane select the Source Statistics tab. It may take a moment for the statistics to appear. The statistics include the size of the table, the number of rows and columns, the column names, data types, number of distinct values, and other information. Below the details about the columns is a bar graph that displays the top unique values for the selected column.

Preview Target Data

For all options except Create Table, to view the existing data in the target table, in the settings pane select the Target Table tab. The target preview displays the data in the table before you run the data load job.

Run the Data Load Job

When you have added all of the source tables for the job and specified the settings for each table, to run the job click the Start icon in the Data Load Cart menu bar. In the Run Data Load Job dialog box, click Run. To stop the data load job, click the Stop icon.

At the top of the page, the Status shows the number of items for which the load has completed over the number of items in the job, and the total time elapsed for the job. When the data load job completes, the Load Tables page displays the results of the job.
To view information about an item in the job, click the Settings icon in the item. The settings pane has the same **Settings**, **Source Table**, **Source Statistics**, **Target Table**, **SQL** and **Errors** tabs as the settings pane before running the job, except that the target preview now contains the data loaded by the data load job. To close the settings pane, click **Close**.

To view a log of the load operation, click the Log icon. You can save the log, clear it, or refresh it. Click **OK** to dismiss the log.

The list of tables on the Data Load / Explore page contains any new tables created. The target tables for the **Insert into Table**, **Replace Data**, **Drop Table and Create New Table**, and **Merge into Table** options contain the loaded data.

Click the **Explore Catalog** button on the Local files page to view the new or updated table on the Catalog page.

**Fixing a data load job**

After your data load job, you might see errors that you want to correct, or upon inspection, realize that you wanted to name a column differently. In such cases, click the **Reload Cart** option to reload cards from your recent cart and edit them as you did before your first attempt. Click the pencil icon to make any changes to the data load job (i.e., change a column name).

Click **Done** to return to the Database Actions page.

---

**Loading Data from Cloud Storage**

You can load data from a cloud store to a table in your Autonomous Database.

You can load files in these file formats: AVRO, CSV, JSON, ORC, Delimited TXT, GZ, GNU ZIP and Tab-Separated Values. For information on supported file formats, see Format Specifications for JSON and AVRO Files.

You can set filters on the data for a table to load only the specified data. For example, to limit the files to only those that are CSV files, enter `*.CSV` in the file extension filter.

Configure and run a data load job from the Load Cloud Object page. To open that page:

1. Open the Database Actions **Data Load page**.
2. Select **LOAD DATA** and **CLOUD STORAGE**, and then click **Next**.

On the left side of the page is a **navigator pane**, where you choose a cloud store connection and the folders or files containing the data. On the right of the page is the data load **cart**, where you stage the files and folders for the data load job. You can set options for the data load job before running it. The Autonomous Database comes with predefined CPU/IO shares assigned to different consumer groups. You can set the consumer group to either low, medium or high while executing a data load job depending on your workload.

To load files from a cloud store into your database, do the following:

- Manage Cloud Storage Links for Data Load Jobs
- Prepare the Data Load Job
- Add Files or Folders for the Data Load Job
- Enter Details for the Data Load Job
- Run the Data Load Job
- View Details About the Data Load Job After It Is Run

---

The page also includes additional sections on managing cloud storage links for data load jobs, preparing the data load job, adding files or folders, entering details for the data load job, and running the data load job. Each section provides detailed instructions and options available for managing data load jobs in the Autonomous Database environment.
Manage Cloud Storage Links for Data Load Jobs

Before you can load data from a cloud store, you must establish a connection to the cloud store you want to use.

On the Load Cloud Object page:

1. Click Add Cloud Storage at the top of the navigation pane.
2. Enter your information in the Add Cloud Storage pane. See Managing Cloud Storage Connections.
3. Click Test to test the connection and then click Create to create the link and close the pane.

Alternatively,

• On the Data Load page, select CLOUD LOCATIONS, and then click Next to go to the Manage Cloud page.
• On the Link Cloud Object page, click the Manage Cloud Store button at the top of the page to go to the Manage Cloud page.

See Managing Cloud Storage Connections.

To return to the Load Cloud Object page, click Data Load in the breadcrumbs at the top of the page and then navigate back to the page.

Prepare the Data Load Job

As you'll see below in Enter Details for the Data Load Job, the first decision you'll make when configuring a data load job is how to load the source data into a new or existing table in the database. The choices are:

• Create a table and insert data loaded from the source into the new table.
• Insert data loaded from the source into an existing table.
• Delete all data in an existing table and then insert new data from the source into the table.
• Drop a table, create a new table, and then insert data loaded from the source into the new table.
• Merge data from the source into a table by updating existing rows in the table and inserting new rows into the table.

You may have to adjust your source data or your target table so that the source data loads correctly into the external target table. The number, order, and data types of columns in the source must match those in the target. Consider:

• If you're creating a new table or if the columns in your source exactly match the columns in an existing target, you don't have to do any special preparation.
• If the columns in your source don't match the columns in an existing target, you must edit your source files or target table so they do match.
• If you're loading multiple files, you must make sure that:
  – All the source files are of the same type, for example, CSV, JSON, etc.
The number, order, and data types of the columns in all the source files match (and that they match the target, if you're loading into an existing table).

- If you want to partition by date:
  - The source file must contain data where the data type is date or timestamp.
  - You must load a folder containing two or more data sources.
  - The names of the files in the folder must indicate a date or dates, for example, MAR-1999.csv or 2017-04-21.xlsx.

Add Files or Folders for the Data Load Job

Add files from the cloud store to the data load cart, where you can edit the details of the data load job. To add the files:

1. From the list at the top of the navigator pane on the left, select the bucket with your source data.
   
   The list shows links that were established on the Manage Cloud Storage page. If you haven't yet registered the cloud store you want to use, click the Manage Cloud Store button at the top of the page and register a connection. See Manage Cloud Storage Links for Data Load Jobs, above.

2. Drag one or more items from the file navigator on the left and drop them into the cart on the right.
   - You can add files, folders, or both. A card is added to the cart for each file or folder you drag into it. The card lists the name of the source file or folder and a proposed name for the target table.
   - If you add a folder that contains multiple files, all the files must be of the same type, that is, CSV, TXT, etc.
   
   When you add the folder to the cart, a prompt is displayed that asks if you want to load all the objects from the multiple source files into a single target table. Click OK to continue or Escape to cancel.
   - When you add multiple individual files or multiple folders to the cart, the data represented by each card will be loaded into a separate table, but all the items in the cart will be processed as part of the same data load job.
   - You can add files or folders from a different bucket, but if you do that, you're prompted to remove all files that are already in the cart before proceeding. To select files from a different bucket, select the bucket from the drop-down list in the navigator pane on the left and then add the file(s), as described above.
   - You can drop files or folders into the data load cart and then navigate away from the Data Load Object page. When you return to the page, those items remain on the page, but you may receive a message, "Remove All Data Load Items. Changing to another Cloud storage location requires all items to be removed from the data load job. Do you wish to continue?" Click Yes to remove the items from the cart. Click No to keep the items in the cart. Then you can continue to work.

You can remove items from the cart before running the data load job:

- To remove an item from the cart, click Remove on the card for the item.
• To remove all items from the cart, click **Remove All** in the data link cart menu bar at the top of the pane.

### Enter Details for the Data Load Job

Enter the details about the data load job in the Load Data from Cloud Storage pane.

On the card in the data link cart, click **Settings** to open the Load Data from Cloud Storage pane for that job. The pane contains:

- **Settings Tab - Table Section**
- **Settings Tab - Properties Section**
- **Settings Tab - Mapping Section**
- **File Tab**
- **Table Tab**
- **Error Tab**
- **Close Button - Save and Close the Pane**

#### Settings Tab - Table Section

Set details about the target table in the **Table** section.

- **Option**: Select an item from the **Option** list to specify how the data should be loaded into a new or existing table. The processing options are:
  - **Create Table**: Creates a table and inserts the data into the new table. When you select this option, the **Name** field on the **Settings** tab is filled with a default name, based on the name of the source file or folder. You can change it if you want.
  - **Insert into Table**: Inserts data loaded from the source into an existing table. When you select this option, the **Name** field on the **Settings** tab presents a list of the tables in the current schema. Select the table into which you want to insert the data.
  - **Replace Data**: Deletes all data in the existing table and then inserts new data from the source into the table. When you select this option, the **Name** field on the **Settings** tab presents a list of the tables in the current schema. Select the table you want to use.
  - **Drop Table and Create New Table**: Drops the table (if it already exists), creates a new table, and then inserts the new data into the table. When you select this option, the **Name** field on the **Settings** tab presents a list of the tables in the current schema. Select the table you want to use.
  - **Merge into Table**: Updates existing rows and inserts new rows in the table. When you select this option, the **Name** field on the **Settings** tab presents a list of the tables in the current schema. Select the table you want to use.

- **Name**: The name of the target table.

- **Partition Column**: List Partitions and Date-based partitions are the different types of partitions available in data loading.
List partitioning is required when you specifically want to map rows to partitions based on discrete values.

To partition according to a specific column, click the **Partition Column** drop-down list and select the column you want to use for the partitioning.

You will have N files per partition value, all partitioned by the partition column you select.

---

**Note:**

- For linked files (from external tables) there is also a requirement that for each file, the list partitioning column can contain only a single distinct value across all of the rows.
- If a file is list partitioned, the partitioning key can only consist of a single column of the table.

---

Date-based partitioning is available when you load a folder containing two or more data sources that contain date or timestamp data.

To partition according to date, click the **Partition Column** drop-down list and select the **DATE** or **TIMESTAMP** column you want to use for the partitioning.

---

### Settings Tab - Properties Section

Specify options to control how the source data is interpreted, previewed, and processed. These options vary, depending on the type of source data.

- **Encoding:** Select a character encoding type from the list. This option is available when the loaded file is in plain text format (CSV, TSV, or TXT). The default encoding type is UTF-8.

- **Text enclosure:** Select the character for enclosing text: " (double-quote character), ' (single-quote character) or **None**. This option is visible only when the selected file is in plain text format (CSV, TSV, or TXT).

- **Field delimiter:** Select the delimiter character used to separate columns in the source. For example, if the source file uses semicolons to delimit the columns, select **Semicolon** from this list. The default is **Comma**. This option is visible only when the selected file is in plain text format (CSV, TSV, or TXT).

- **Rows to Skip:** Specifies the number of rows to skip when loading the source data into the target:
  - If you select the **Get from file header** option under **Source column name** (see below) and if you enter a number greater than 0 in the **Rows to skip** field, then that number of rows after the first row are not loaded into the target.
  - If you deselect the **Get from file header** option under **Source column name**, and if you enter a number greater than 0 in the **Rows to skip** field, then that number of rows including the first row are not loaded into the target.

- **Source column name:** Select the **Get from file header** checkbox to use the column names form the source table in the target table.
– If you select this option, the first row in the file is processed as column names. The rows in the Mapping section, below, are filled with those names (and with the existing data types, unless you change them).
– If you deselect this option, the first row is processed as data. To specify column names manually, enter a name for each target column in the Mapping section. (You will also have to enter data types.)

• **Numeric column**: Select the **Convert invalid data to null** checkbox to convert an invalid numeric column value into a null value.

**Settings Tab - Mapping Section**

The settings in the Mapping section control how data from the source files are loaded into the rows of the target database table. For each row, the data from the column listed under **Source column** will be loaded into the column listed under **Target column**.

As mentioned above, the contents of the Mapping table change according to what processing option you chose in the Table section and which properties you set in the Properties section.

• **Source column**: Lists the columns from the source file.
  
  If the **Get from file header** option under Properties is selected, **Source column** shows the names of the columns in the source file. If the **Get from file header** option is not selected, generic names like **COLUMN_1**, **COLUMN_2**, etc., are used. This field is always read only.

• **Target column**: Lists the columns in the target table. Accept, select, or enter a column in the target table.

  The contents of this column differ, depending on what you selected for the table processing **Option** and whether you selected for the **Get from file header** option.

  – If (1) the processing option is **Create Table** or **Drop Table and Create New Table** and (2) the **Get from file header** option is selected, then the **Target column** uses the names of the columns in the source file. You can change the name of a target column by replacing the provided name with a new one.

  – If (1) the processing option is **Create Table** or **Drop Table and Create New Table** and (2) the **Get from file header** option is not selected, then generic names like **COLUMN_1**, **COLUMN_2**, etc., are used. You can change the name of a target column by replacing the provided name with a new one.

  – If (1) the processing option is **Insert into Table**, **Replace Data**, or **Merge Into Table** and (2) the **Get from file header** option is selected, then the **Target column** has a drop-down list of all the columns in the target table, with their data types. By default, the column with the name corresponding to the source column is selected, but you can select a different one from the list.

  – If (1) the processing option is **Insert into Table**, **Replace Data**, or **Merge Into Table** and (2) the **Get from file header** option is not selected, then the **Target column** has a drop-down list of all the columns in the target table, with their data types. Select a column from the list to use as the target column.
Data Type: Lists the data type to use for data in that column. This column is displayed only for Create Table or Drop Table and Create New Table. The contents change depending on whether the Get from file header option is selected.

- If the Get from file header option is selected, Data type shows the data types of the columns in the source file (for Create Table) or in the existing table (for Drop Table and Create New Table). If you want to change the data type for the target, click the name and select a different one from the list.
- If the Get from file header option is not selected, Data type shows all available data types. Select the data type to use for the target column from the list.

Length/Precision (Optional): For columns where the Data Type is NUMBER, enter the length/precision for the numbers in the column. Precision is the number of significant digits in a number. Precision can range from 1 to 38.

For columns where Data Type is VARCHAR2, the Auto value in Length/Precision field enables the Auto Size feature.

With the Auto-Size column Width feature, you can automatically size any column to fit the largest value in the column. Select Auto from the Length/Precision drop-down values or pick a value from the drop-down list.

Scale (Optional): For columns where the Data Type is NUMBER, enter the scale for the numbers in the column. Scale is the number of digits to the right (positive) or left (negative) of the decimal point. Scale can range from ranges from -84 to 127.

Format: If the data type in the Data type column is DATE or one of the TIMESTAMP types, select a format for that type from the from the Format drop-down list.

Merge Key: This option is used only for the processing option Merge into Table.

For the Merge into Table option, you must specify at least one column to use as a key column. Merge keys are one or more columns that uniquely identify each row in the table. To specify a key column, select the Merge Key checkbox for the column. Merge keys must not contain any null values. For loading tables with primary keys, this option automatically enables the selection of primary key columns as the merge keys.

File Tab

The File tab displays the source data in tabular form. The display reflects the settings you chose in the Properties section.

If you dragged a folder containing multiple files into the data load cart and then clicked Settings for that card, the File pane includes a Preview Object (File) drop-down list at the top of the pane that lists all the files in the folder. Select the source file you want to preview from that list.

Table Tab

The Table tab displays what the target table is expected to look like after the data has been loaded. If you chose the Create Table processing option, no table is shown.
Errors Tab

The **Errors Tab** lists any errors generated when attempting to run the data load job.

SQL Tab

The **SQL Tab** displays the SQL commands that will be run to complete this data load job.

![Note:](Note: You can see the SQL code even before the table is created.)

Close Button - Save and Close the Pane

After entering all the details for the data load job, click **Close** at the bottom of the page. This saves the details you entered and returns you to the Load Data from Cloud Storage pane. To close the page without saving your entries, press **Escape**.

Run the Data Load Job

Once you've added data sources to the data load cart and entered details about the data load job, you can run the job.

To run the job:

1. If you haven't already done so, click the **Close** button in the **Load Data from Cloud Storage** pane to save your settings and close the pane. If any of the settings are invalid, an error message reports the problem. Fix the problem and click **Close**.

2. Click **Start** in the data load cart menu bar. To stop the data load job, click **Stop**.

When the data load job completes, the Load Cloud Object page displays the results of the job. At the top of the page, a **Status** message shows the number of items for which the load has completed over the number of items in the job and the total time elapsed for the job.

View Details About the Data Load Job After It Is Run

To view details about the data load job after it is run, click **Settings** in the card for the item. The **Load Data from Cloud Storage** pane is displayed again, with the settings used for the job plus some additional details about job run.

Settings Tab

The **Settings** tab shows the details that were set in the **Settings** tab when preparing for job.
File Tab
The File tab shows the source file or files used for the data load job.

Table Tab
The Table tab shows the table created or modified from the data load job.

Error Tab
The Error tab shows rows rejected from the data link job, if any. You can click the icon at the top of the page to download the rejected rows as a CSV file.

SQL Tab
The SQL tab shows the SQL that was created and run to load the data. You can click the Copy to clipboard button at the top of the pane to copy it.

Data Definition Tab
The Data Definition tab shows the data definition of the table created from the data load job.

View the Table Resulting from the Data Load Job
To view the new tables or tables modified by the data load job, you can:

1. Fix your data load job. After your data load job, you might see errors that you want to correct, or upon inspection, realize that you wanted to name a column differently. In such cases, click the Reload Cart option to reload cards from your recent cart and edit them as you did before your first attempt. Click the pencil icon to make any changes to the data load job (i.e., change a column name).

2. On the Load Cloud Object page, click the Explore Catalog button.

3. On the Catalog page, find the new or updated table.

4. Click View Details on the right side of the card and review the table. See The Catalog Page.

Alternatively, you can:

1. From the Database Actions Data Load page, click Explore.

2. On the Explore page, click the name of the table you want to review. See Exploring Data.

Linking Data
You can link to data in remote databases or in cloud storage buckets.

When you link to data in a remote database or in cloud storage, the target object produced is an external table or a view. When you select that target table or view on the Data Load - Explore page, the source preview for the object shows the current data in the source object.

Linking to columns in a remote database or to files in cloud storage can be useful when the source data is being continually updated. For example, if you have a database table with
columns that are updated for each new sales transaction, and you have run a data
load job that links columns of the source table to targets in your Oracle Autonomous
Database, then those targets have the new sales data as it is updated in the source.

See:

• Linking to Other Databases
• Linking to Objects in Cloud Storage

Linking to Other Databases

To link to data in tables in another database from your Oracle Autonomous Database,
on the Data Load page, select LINK DATA and DATABASE, then click Next. Select a
database from the drop-down list. Drag one or more tables from the list of database
tables and drop them in the Data Load Cart.

Each table appears as an item in the Data Load Cart. The item shows the name of the
table and the number of rows in it, and the name of the table that is the target for the
data load.

To remove a table from the Data Load Cart, click the Remove (trash can) icon for the
item. To remove all tables from the cart, click the Remove All (trash can) icon in the
Data Load Cart menu bar.

To add a remote database to the list of databases, create a database link to the
remote database. For information on creating a database link, see Database Links in
Oracle® Database Database Administrator’s Guide.

The databases available to you appear in the drop-down list of the database
navigation pane of the Link Tables page.

You can filter the tables displayed in the navigation pane by entering a case-sensitive
value in the search field at the top of the navigation tree and pressing Enter. To display
all of the tables again, clear the search field and press Enter.

You can add any number of tables from the navigation pane to the Data Load Cart and
create links to each of them in a single data loading job. You can set filters on the data
for a table to load only the specified data. When you run the data load job, a view to
each table is created in your Oracle Autonomous Database.

Specify the Target, Create Filters, and View Mappings

To specify settings for the data load job, preview the data in the source or the target,
and see statistics about the data, click the Settings (pencil) icon for the item in the
Data Load Cart.

In the settings pane, on the Settings tab, the Source field displays the name of the
table and the number of rows in the table.

The Target field specifies the name of the target view. To specify a different name for
the target, enter it in the Target field.

You can set filters to load only the specified data from the source table. To set an initial
filter condition on a table, do the following:

1. Click the Edit Source Filter icon.
2. In the Edit Source Filter dialog box, click Add Filter Condition.
3. In the Add Filter Condition dialog box, select a source column and an operator and specify a value.

4. Click Save.

To add another filter condition, click the Edit Source Filter icon and repeat the steps for adding a filter. The filter then has both filter conditions and the AND operator. You can change the AND to an OR by clicking the Edit Source Filter icon and manually replacing the AND with OR in the filter field.

To remove a filter, in the Edit Source Filter dialog box, delete the value in the Filter field of the Edit Source Filter dialog box.

The Mapping section displays the columns in the source table and those in the target view.

Preview Source Data

To view the data in the source table, in the settings pane select the Source Preview tab. The source preview displays the data in the table.

View Statistics

To view statistics about the source table, in the settings pane select the Statistics tab. It may take a moment for the statistics to appear. The statistics include the size of the table, the number of rows and columns, the column names, data types, number of distinct values, and other information. Below the details about the columns is a bar graph that displays the top unique values for the selected column.

Preview Target Data

To view the data in the target view, in the settings pane select the Target Preview tab. The target preview displays the data in the target view. If the view does not yet exist, then the target data is the same as the source data, regardless of any filters set for the data load job.

Run the Data Load Job

When you have added all of the source tables for the job and specified the settings for each table, to run the job click the Start icon in the Data Load Cart menu bar. In the Run Data Load Job dialog box, click Run. To stop the data load job, click the Stop icon.

At the top of the page, the Status shows the number of items for which the load has completed over the number of items in the job, and the total time elapsed for the job. When the data load job completes, the Link Tables page displays the results of the job.

To view information about an item in the job, click the Settings (circled i) icon in the item. The settings pane has the same Settings, Source Preview, Statistics, and Target Preview tabs as the settings pane before running the job, except that the Target Preview now contains the data loaded by the data load job. To close the settings pane, click Close.

To view a log of the load operation, click the Logging icon. You can save the log, clear it, or refresh it. Click OK to dismiss the log.

The list of views on the Data Load / Explore page contains any new views created. A preexisting view that was the target for the data load job now contains the loaded data.

On the Database links page, click the Reload Cart button. Clicking the button enables you to add the data sources to the data load the cart again.

Click Done to return to the Database Actions page.
Linking to Objects in Cloud Storage

When you create a link to files in a cloud store bucket from your Oracle Autonomous database, you create an external table that links to the files in the cloud store.

You can link to files in these file formats: AVRO, CSV, JSON, Parquet, ORC, Delimited TXT. For information on supported file formats, see Format Specifications for JSON, AVRO, and XML Files.

Configure and run a data link job from the Link Cloud Object page. To open that page:

1. Open the Database Actions Data Load page.
2. Select LINK DATA and CLOUD STORAGE, and then click Next.

On the left side of the page is a navigator pane, where you choose a cloud store connection and the folders or files containing the data. On the right of the page is the data load cart, where you stage the files and folders for the data link job. You can set options for the data link job before running it. The Autonomous Database comes with predefined CPU/I/O shares assigned to different consumer groups. You can set the consumer group to either low, medium or high while executing a data load job depending on your workload.

To link to files from a cloud store, do the following:

- Manage Cloud Storage Links for Data Link Jobs
- Prepare the Data Link Job
- Add Files or Folders for the Data Link Job
- Enter Details for the Data Link Job
- Run the Data Link Job
- View Details About the Data Link Job After It Is Run
- View the Table Resulting from the Data Link Job

Manage Cloud Storage Links for Data Link Jobs

Before you can link to data in a cloud store, you must establish a connection to the cloud store you want to use.

On the Link Cloud Object page:

1. Click Add Cloud Storage at the top of the navigation pane.
2. Enter your information in the Add Cloud Storage pane. See Managing Cloud Storage Connections.
3. Click Test to test the connection and then click Create to create the link and close the pane.

Alternatively,

- On the Data Load page, select CLOUD LOCATIONS, and then click Next to go to the Manage Cloud page.
- On the Load Cloud Object page, click the Manage Cloud Store button at the top of the page to go to the Manage Cloud page.
See Managing Cloud Storage Connections.

To return to the Link Cloud Object page, click Data Load in the breadcrumbs at the top of the page and then navigate back to the page.

Prepare the Data Link Job

You may have to adjust your source data or your target table so that the source data links correctly to the external target table. Consider:

- If you’re linking to multiple files, you must make sure that:
  - All the source files are of the same type, for example, CSV, JSON, etc.
  - The number, order, and data types of the columns in all the source files match.
- If you want to partition by date:
  - The source file must contain data where the data type is date or timestamp.
  - You must load a folder containing two or more data sources.
  - The names of the files in the folder must indicate a date or dates, for example, MAR-1999.csv or 2017-04-21.xlsx.

Add Files or Folders for the Data Link Job

Add files from the cloud store to the data link cart, where you can edit the details of the data link job. To add the files:

1. From the list at the top of the navigator pane on the left, select the bucket with your source data.

   The list shows links that were established on the Manage Cloud Storage page. If you haven’t yet registered the cloud store you want to use, click the Manage Cloud Store button at the top of the page and register a connection. See Manage Cloud Storage Links for Data Link Jobs, above.

2. Drag one or more items from the file navigator on the left and drop them into the cart on the right.

   - You can add files, folders, or both. A card is added to the cart for each file or folder you drag into it. The card lists the name of the source file or folder and a proposed name for the target table.
   - If you add a folder that contains multiple files, all the files must be of the same type, that is, CSV, TXT, etc.

   When you add the folder to the cart, a prompt is displayed that asks if you want to load all the objects from the multiple source files into a single target table. Click OK to continue or Escape to cancel.

   - When you add multiple individual files or multiple folders to the cart, the data represented by each card will be loaded into a separate table, but all the items in the cart will be processed as part of the same data load job.

   - You can add files or folders from a different bucket, but if you do that, you’re prompted to remove all files that are already in the cart before proceeding. To select files from a different bucket, select the bucket from the drop-down list in the navigator pane on the left and then add the file(s), as described above.
• You can drop files or folders into the data load cart and then navigate away from the Data Link Object page. When you return to the page, those items remain on the page, but you may receive a message, “Remove All Data Link Items. Changing to another Cloud storage location requires all items to be removed from the data load job. Do you wish to continue?” Click Yes to remove the items from the cart. Click No to keep the items in the cart. Then you can continue to work.

You can remove items from the cart before running the data link job:

• To remove an item from the cart, click Remove on the card for the item.

• To remove all items from the cart, click Remove All in the data link cart menu bar at the top of the pane.

Enter Details for the Data Link Job

Enter the details about the data link job in the Link Data from Cloud Storage pane.

On the card in the data link cart, click Settings to open the Link Data from Cloud Storage pane for that job. The pane contains:

• Settings Tab - Table Section
• Settings Tab - Properties Section
• Settings Tab - Mapping Section
• File Tab
• Table Tab
• Error Tab
• SQL Tab
• Close Button - Save and Close the Pane

Settings Tab - Table Section

Set details about the target table in the Table section.

• Name: The name of the target table.

• Partition Column:

List Partitions and Date-based partitions are the different types of partitions available in data linking.

List partitioning is required when you specifically want to map rows to partitions based on discrete values.

To partition according to a specific column, click the Partition Column drop-down list and select the column you want to use for the partitioning.

You will have N files per partition value, all partitioned by the partition column you select.
Date-based partitioning is available when you link a folder containing two or more data sources that have columns that contain date or timestamp data.

To partition according to date, click the **Partition Column** drop-down list and select the **DATE** or **TIMESTAMP** column you want to use for the partitioning.

- **Validation Type:** Validation examines the source files, optional partitioning information, and report rows that do not match the format options specified. Select **None** for no validation; select **Sample** to perform validation based on a sample of the data; or select **Full** to perform validation based on all the data.

- **Use Wildcard:** This check box enables use of wildcard characters in search condition to retrieve specific group of files that matches the filter criteria. You can use a wildcard character, such as an asterisk (*) that searches, filters, and specifies groups of files that detect and add new files to the external table.

  For example, if you enter file*, then file01, file02, file03, and so on are considered to match the keyword. The asterisk (*) matches zero or more characters of the possibilities, to the keyword.

**Note:**

The wildcard support is incompatible with partitioning. The validation of source file fails if you use wildcards with partitioned data.

**Settings Tab - Properties Section**

Specify options to control how the source data is interpreted, previewed, and processed. These options vary, depending on the type of source data.

- **Encoding:** Select a character encoding type from the list. This option is available when the linked file is in plain text format (CSV, TSV, or TXT). The default encoding type is **UTF-8**.

- **Text enclosure:** Select the character for enclosing text: " (double-quote character), ' (single-quote character) or **None**. This option is visible only when the selected file is in plain text format (CSV, TSV, or TXT).

- **Field delimiter:** Select the delimiter character used to separate columns in the source. For example, if the source file uses semicolons to delimit the columns, select **Semicolon** from this list. The default is **Comma**. This option is visible only when the selected file is in plain text format (CSV, TSV, or TXT).

- **Rows to Skip:** Specifies the number of rows to skip when linking the source data to the target external table:
— If you select the **Get from file header** option under **Source column name** (see below) and if you enter a number greater than 0 in the **Rows to skip** field, then that number of rows after the first row are not linked to the target.

— If you deselect the **Get from file header** option under **Source column name**, and if you enter a number greater than 0 in the **Rows to skip** field, then that number of rows including the first row are not linked to the target.

**Source column name:** Select the **Get from file header** checkbox to use the column names form the source table in the target table.

— If you select this option, the first row in the file is processed as column names. The rows in the **Mapping** section, below, are filled with those names (and with the existing data types, unless you change them).

— If you deselect this option, the first row is processed as data. To specify column names manually, enter a name for each target column in the **Mapping** section. (You will also have to enter data types.)

**Numeric column:** Select the **Convert invalid data to null** checkbox to convert an invalid numeric column value into a null value.

### Settings Tab - Mapping Section

The settings in the **Mapping** section control how data from the source files are linked to the rows of the target external table. For each row, the data from the column listed under **Source column** will be linked to the column listed under **Target column**.

**Source column:** Lists the columns from the source file.

If the **Get from file header** option under **Properties** is selected, **Source column** shows the names of the columns in the source file. If the **Get from file header** option is not selected, generic names like COLUMN_1, COLUMN_2, etc., are used. This field is always read only.

**Target column:** Lists the columns in the target table.

— If the **Get from file header** option is selected, then the **Target column** uses the names of the columns in the source file. You can change the name of a target column by replacing the provided name with a new one.

— If the **Get from file header** option is not selected, then generic names like COLUMN_1, COLUMN_2, etc., are used. You can change the name of a target column by replacing the provided name with a new one.

**Data Type:** Lists the data type to use for data in that column. The contents change depending on whether the **Get from file header** option is selected.

— If the **Get from file header** option is selected, **Data type** shows the data types of the columns in the source file. If you want to change the data type for the target, click the name and select a different one from the list.

---

**Note:**

If you're linking multiple files from a folder in a single data link job, only the first file will be shown in the **Mapping** section. However, as long as the column names and data types match, the data from all source files will be linked.
If the Get from file header option is not selected, Data type shows all available data types. Select the data type to use for the target column from the list.

- Length/Precision (Optional): For columns where the Data Type is NUMBER, enter the length/precision for the numbers in the column. Precision is the number of significant digits in a number. Precision can range from 1 to 38.

For columns where Data Type is VARCHAR2, the Auto value in Length/Precision field enables the Auto Size feature.

With the Auto-Size column Width feature, you can automatically size any column to fit the largest value in the column. Select Auto from the Length/Precision drop-down values or pick a value from the drop-down list.

- Scale (Optional): For columns where the Data Type is NUMBER, enter the scale for the numbers in the column. Scale is the number of digits to the right (positive) or left (negative) of the decimal point. Scale can range from ranges from -84 to 127.

- Format: If the data type in the Data type column is DATE or one of the TIMESTAMP types, select a format for that type from the from the Format drop-down list.

File Tab
The File tab displays the source data in tabular form. The display reflects the settings you chose in the Properties section.

If you dragged a folder containing multiple files into the data link cart and then clicked Settings for that card, the File pane includes a Preview Object (File) drop-down list at the top of the pane that lists all the files in the folder. Select the source file you want to preview from that list.

Table Tab
The Table tab displays what the target table is expected to look like after the data has been linked.

Error Tab
The Error tab lists any errors generated when attempting to run the data link job.

SQL Tab
The SQL tab displays the SQL commands that will be run to complete this data link job.

Note:
You can see the SQL code even before the table is created.

Close Button - Save and Close the Pane
After entering all the details for the data link job, click Close at the bottom of the page. This saves the details you entered and returns you to the Link Data from Cloud Storage pane. To close the page without saving your entries, press Escape.
Run the Data Link Job

Once you’ve added data sources to the data link cart and entered details about the data link job, you can run the job.

To run the job:

1. If you haven’t already done so, click the Close button in the Link Data from Cloud Storage pane to save your settings and close the pane. If any of the settings are invalid, an error message reports the problem. Fix the problem and click Close.

2. Click Start in the data link cart menu bar. To stop the data link job, click Stop.

When the data link job completes, the Link Cloud Object page displays the results of the job. At the top of the page, a Status message shows the number of items for which the link has completed over the number of items in the job and the total time elapsed for the job.

View Details About the Data Link Job After It Is Run

To view details about the data link job after it is run, click Settings in the card for the item. The Link Data from Cloud Storage pane is displayed again, with the settings used for the job plus some additional details about job run.

Settings Tab

The Settings tab shows the details that were set in the Settings tab when preparing for job.

File Tab

The File tab shows the source file or files used for the data link job.

Table Tab

The Table tab shows the external table created for the data link.

Error Tab

The Error tab shows rows rejected from the data link job, if any. You can click the icon at the top of the page to download the rejected rows as a CSV file.

SQL Tab

The SQL tab shows the SQL that was created and run to link the data. You can click the Copy to clipboard button at the top of the pane to copy it.

View the Table Resulting from the Data Link Job
After running a data link job, the Link Cloud Object page is shown, with an **Explore Catalog** button on the bottom of the page. To view the table created by the data link job:

1. On the Database links page, click the **Reload Cart** button. Clicking the button enables you to add the data sources to the data load the cart again.
2. Click the **Explore Catalog** button.
3. On the Catalog page, find the new or updated table.
4. Click **View Details** on the right side of the card and review the table. See The Catalog Page.

Alternatively, you can:

1. From the Database Actions Data Link page, click **Explore**.
2. On the Explore page, click the name of the table you want to review. See Exploring Data.

### Feeding Data

You can run a live table feed on demand, on a schedule, or as the result of a notification.

The bucket can contain files in these formats: AVRO, CSV, JSON, Parquet, ORC, Delimited TXT. All of the files must have the same column signature.

#### About the Live Feed Page

On the Database Actions - Data Load page select **FEED DATA** and **CLOUD STORAGE**, then click **Next** to display the Live Feed page. On this page, you can:

- Manage Cloud Storage Connections for Live Table Feeds
- Create a Live Table Feed Object
- List, Filter, and Sort Live Table Feed Objects
- Find and View Live Table Feed Objects
- Run a Live Table Feed
- Suspend or Resume a Live Table Feed Job
- Delete a Live Table Feed

#### Manage Cloud Storage Connections for Live Table Feeds

Before you create a live table feed, you must establish a connection to the cloud store you want to use:

1. Click the **Manage Cloud Store** button at the top of the page to go to the Manage Cloud page. For instructions, see Managing Cloud Storage Connections.
2. To return to the Live Feed page, click **Data Load** in the breadcrumbs at the top of the page and then navigate to the Live Feed page, as described in About the Live Feed Page, above.

#### Create a Live Table Feed Object

To create a live table feed object,

1. On the Live Feed page, click the **Create Live Table Feed** button to display the Create Live Feed pane. Enter information as follows:
• **Live Table Feed Name**: Accept the default name or enter a different name to identify this live table feed.

• **Target Table Name**: Accept the default name or enter a different name. This is the name of the target table that will be created in your Autonomous Database instance to store the data from the live feed.

• **Object Filter (Regular Expression)**: Enter a regular expression to limit the live table feed to only those files in the bucket that match the expression. For example, to limit the files to only those that are CSV files with names that start with `SALES`, enter `SALES\.\*\.CSV`

• **Schema**: Select the schema you want to use from the drop-down list.

• **Cloud Storage**: Select the cloud connection for the bucket containing the file you want to use for feeding data.

Connections that were established on the Manage Cloud page are listed here. If you haven’t registered the cloud store you want to use yet, click the **Manage Cloud Store** button at the top of the page and register a connection to a bucket in a cloud store. See **Managing Cloud Storage Connections**.

• **Enable for Notification**: Select this option so that new or changed data in the data source will be loaded based on an Oracle Cloud Infrastructure notification. When you select this option, you can avoid delays that might occur when polling is initiated on a schedule (that is, if you selected the live table feed **Scheduled** option).

When you select the **Enable for Notification** option, you must also:

– Configure your object store bucket to emit notifications
– Create a Notifications service subscription topic
– Create an Events service rule
– Copy the notification URL
– Create a Notifications service subscription
– Confirm that notifications are allowed

For complete instructions, see **Creating a Notification-Based Live Table Feed**.

• **Scheduled**: Select this option to set up a schedule for running the live table feed object; that is, to poll the data source on a regular basis:

– In the time interval fields, enter a number, and select a time type and the days on which to poll the bucket for new or changed files. For example, to poll every two hours on Monday, Wednesday, and Friday, enter **2**, select **Hours**, and then select **Monday, Wednesday, Friday** in the appropriate fields.

– Select a start and end date. If you don’t select a start date, the current time and date are used as the start date. The end date is optional. However, without an end date, the live feed will continue to poll.

2. Click **Create**.

**List, Filter, and Sort Live Table Feed Objects**

When you open the Live Feed page, existing live table feed objects are displayed as cards on the page. They are identified as **LIVE_TABLE_FEED entities**.

**To filter live table feed objects:**
1. Click the **Display or hide filter panel** (funnel) icon at the top left of the page to display filter options. By default, the live table feed objects from the current user’s schema are shown.

2. To include objects from other schemas, select the check boxes next to the names of the schemas you want, under **Schemas**. To remove a schema from the filter list, deselect the box next to its name.

3. To show objects from all available schemas, select all the schemas in the list or deselect all the schemas in the list.

**To sort live table feed objects**

1. Click the **Sort by** button at the top right of the page.

2. Select a sorting option. To sort ascending, click the icon with the up arrow. To sort descending, click the icon with the down arrow.

**Find and View Live Table Feed Objects**

To search for available live table feed entities in the selected schemas, enter a value in the search field at the top of the page and press **Enter**. The display then includes only the entities whose names contain the characters in the search field. To clear the search field, click the Clear search results (X) icon in the search field.

To remove a schema or sorting value from the selected filters, deselect the schema or sorting value in the filter panel, or click the Remove filter (X) icon for the schema or sorting value above the display of live table feed objects. To close the filter panel, click the Hide filter panel (X) icon in the panel.

To refresh the display of live table feeds, click the Refresh icon at the top of the page.

**Edit a Live Table Feed Object**

To edit details of a live table feed object,

1. On the Live Feed page, find the card for the live table feed whose details you want to edit.

2. Click the Actions icon (three dots) on the card and select **Edit Live Table Feed**. You can edit the following options:
   - **Object Filter (Regular Expression):** Enter a regular expression to limit the live table feed to only those files in the bucket that match the expression. For example, to limit the files to only those that are CSV files with names that start with SALES, enter `SALES.*.CSV`.
   - **Enable for Notification:** Select this option so that new or changed data in the data source will be loaded based on an Oracle Cloud Infrastructure notification. When you select this option, you can prevent any delays that might occur when polling is initiated on a schedule (that is, the live table feed **Scheduled** option).

   When you select the **Enable for Notification** option, you must also:
   - Copy the live table feeds notification URL
   - Configure your cloud store to emit notifications
   - Configure Oracle Cloud Infrastructure to route events to the endpoint used for the live table feed.
   - Create a rule.
   - Create a subscription.
– Confirm that notifications are allowed at the live feed service.

For complete instructions, see Creating a Notification-Based Live Table Feed.

• **Scheduled:** Select this option to set up a schedule for running the live table feed object; that is, to poll the data source on a regular basis:
  – In the time interval fields, enter a number, and select a time type and the days on which to poll the bucket for new or changed files. For example, to poll every two hours on Monday, Wednesday, and Friday, enter 2, select **Hours**, and then select **Monday, Wednesday, Friday** in the appropriate fields.
  – Select a start and end date.

3. Click **Save**.

### Run a Live Table Feed

You can run a live table feed on demand, on a schedule, or as the result of a notification.

**To run a live table feed on demand:**

1. On the Live Feed page, find the card for the live table feed you want to run.
2. Click the Actions icon (three dots) on the card and select **Run Live Table Feed Immediately (Once)**.

**To run a live table feed on a schedule:**

You can set a schedule for running live table feeds on the **Create Live Table Feed** pane (when creating a new table feed) or the **Edit Live Table Feed** pane (when editing an existing table feed). See **Create a Live Table Feed Object** or **Edit a Live Table Feed Object**.

**To run a live table feed as the result of a notification:**

See Creating a Notification-Based Live Table Feed.

Select the **Scheduled** check box to display the schedule options and then set the schedule by selecting the options you want.

**To view live table feed run details:**

1. On the Live Feed page, find the card for the live table feed whose run details you want to see.
2. Click the Actions icon (three dots) on the card and select **Live Table Feed Run Details**.
   The **Objects** tab on the Live Table Feed Run Details pane displays information about the jobs, such as when the run occurred, the objects involved in the run, the rows loaded and the rows rejected, and other details. Click the **All** tab to view more details, such as the event type.

### Suspend or Resume a Live Table Feed Job

1. On the Live Feed page, find the card for the live table feed job you want to suspend or resume.
2. Click the Actions icon (three dots) on the card and select **Suspend Live Table Feed Job** to suspend the job or **Resume Live Table Feed Job** to resume it.
Delete a Live Table Feed

1. On the Live Feed page, find the card for the live table feed job you want to delete.
2. Click the Actions icon (three dots) on the card and select **Delete Live Table Feed**.

Creating a Notification-Based Live Table Feed

You can load data through a live table feed based on an Oracle Cloud Infrastructure notification.

In addition to being able to run a live table feed on demand or on a schedule, as described in *Feeding Data*, you can also run a feed as the result of a notification. When data in the source bucket is changed, a notification is sent which triggers a run of the table feed. With a notification-based live table feed, you can avoid any delay that might come from running on-demand or scheduled live table feed jobs.

**Note:**

Notification-based live table feeds aren't available on the Oracle Cloud Infrastructure free tier. You must be on a paid tenancy with appropriate permissions on your account to use this feature.

To create a notification-based live table feed:

- **Step 1:** Configure your object store bucket to emit notifications
- **Step 2:** Create a Notifications service subscription topic
- **Step 3:** Create an Events service rule
- **Step 4:** Create and configure a live table feed to use notifications, and copy the notification URL
- **Step 5:** Create a Notifications service subscription
- **Step 6:** Confirm that the endpoint can receive notifications

**Tip:**

To complete those steps, you will alternate between Oracle Cloud Infrastructure Console pages and Oracle Database Actions pages. You may find it convenient to open the Cloud Console in one browser page or tab and Database Actions in another, so it's easy to move back and forth.

**Step 1: Configure your object store bucket to emit notifications**

*Where:* Oracle Cloud Infrastructure Console: Object Storage & Archive Storage - Buckets page

Configure the bucket containing your source data so that it will emit notifications when the data changes. You can set this option when you create a bucket or you can set it in an existing bucket.
1. Open the Cloud Console navigation menu and click Storage. Under Object Storage and Archive Storage, click Buckets.

2. If you're creating a new bucket:
   a. On the Buckets page, click the Create Bucket button to create a new bucket, as described in Managing Buckets. In the Create Bucket wizard, select the Emit Object Events option, along with the other options for your new bucket.
   b. Click Create.

   If you're using an existing bucket:
   a. On the Buckets page, click the name of the bucket you want to use, as described in Managing Buckets.
   b. On the Bucket Details page, click the Edit link next to Emit Object Events.
   c. Select the Emit Objects Events check box, and then click Save Changes.

Step 2: Create a Notifications service subscription topic

Where: Oracle Cloud Infrastructure Console: Notifications - Topics page

1. Open the Cloud Console navigation menu and click Developer Services. Under Application Integration, click Notifications.
2. Click Create Topic, enter a name and optional description, and then click Create.

See also Managing Topics and Subscriptions.

Step 3: Create an Events service rule

Where: Oracle Cloud Infrastructure Console: Events - Rules page

2. Click Create Rule, and fill out the Create Rule page as described in Managing Rules for Events.
   • Under Rule Conditions, select:
     – Condition: Event Type
     – Service Name: Object Storage
     – Event Type: Object - Create
   • Under Actions, select:
     – Action Type: Notifications
     – Notifications Compartment: Select the compartment to use for the notifications.
     – Topic: Select the name of the topic you created above, in Step 2: Create a Notifications service subscription topic.
3. Click Create Rule.

Step 4: Create and configure a live table feed to use notifications, and copy the notification URL

Where: Database Actions: Live Feeds page

You can configure a new or an existing live table feed to use notifications:
1. Go to the Database Actions Live Feeds page, as described in Feeding Data.

2. Create or edit a live table feed object, as described in Create a Live Table Feed Object or Edit a Live Table Feed Object. Select the Enable for Notification option.

3. Click Create or Save.

4. Click the Actions (three vertical dots) icon on the card for your live feed, and select Show Confirmation URL.

5. In the Notification URL dialog box, click the Copy icon to copy the URL to the clipboard. You may want to copy it to a temporary file, so you can retrieve it later. You'll use this URL in the next step, Step 5: Create a Notifications service subscription.

Step 5: Create a Notifications service subscription

Where: Oracle Cloud Infrastructure Console: Notifications - Subscriptions page


2. On the Notifications page, click the Subscriptions tab (on the left side of the page), the status will be Active.

3. Click Create Subscription and fill in the Create Subscription page:
   - Subscription topic: Select the subscription topic you created in Step 2: Create a Notifications service subscription topic.
   - Protocol: HTTPS (Custom URL)
   - URL: Paste in the URL you copied in Step 4: Create and configure a live table feed to use notifications, and copy the notification URL.
   - Click Create. The subscription will be listed in the Subscriptions table in a state of "Pending."

Step 6: Confirm that the endpoint can receive notifications

Where: Database Actions: Live Feeds page

1. Return to the Database Actions Live Feeds page and find the card for the live table feed you are configuring for a notification-based feed.

2. Click the Actions (three vertical dots) icon on the card, and select Show Confirmation URL.

3. In the Confirmation URL dialog box, click the link to confirm the URL. This does not close this dialog box. If the link is successful, a message is displayed that confirms the subscription is active.

4. Return to the Confirmation URL dialog box and select the Check only when the cloud store confirmation process is complete check box, and click OK.

Once you finish the above steps, any new files uploaded to the bucket will automatically be loaded into the live table feed table.

Creating a Notification-Based Live Table Feed using Amazon Simple Storage Service (S3)

You can integrate Amazon Simple Storage Service (S3) and Oracle Cloud Infrastructure (OCI) to automate the process of live feed notifications when storage objects it is observing...
Tip:

To complete these steps, you will need to alternate between Amazon Web Services (AWS) Management console and Oracle Database Actions pages. You may find it convenient to open the Amazon Web Services in one browser page or tab and Database Actions in another, so it is easy to move back and forth.

To create a notification-based live feed with Amazon S3 as cloud storage you must:

- Step 1: Create your object store bucket in Amazon S3
- Step 2: Create Access Keys
- Step 3: Add an OCI Cloud Storage using Amazon S3
- Step 4: Create and configure a live table feed to use notifications and copy the notification URL
- Step 5: Create a notifications service subscription topic
- Step 6: Enable and configure event notifications using the Amazon S3 console
- Step 7: Create a notifications service subscription
- Step 8: Confirm that the endpoint can receive notifications

Step 1: Create your object store bucket in Amazon S3

Where: Amazon Web Services (AWS) Management console

Configure and create your bucket containing source data so that it emits notifications when the data changes.

1. Log in to AWS Management console and open the Amazon S3 console.
2. On the home page click the **Create Bucket** icon.
3. In **Bucket name**, enter a valid name for your bucket. For example: testbucket. After you create the bucket, you cannot change its name.
4. In **Region**, select the Amazon Web Services (AWS) Region from the dropdown. For example: us-west-2
5. In Bucket settings for Block Public Access, select the **Block Public Access** settings that you want to apply to the bucket. It is recommended to keep all settings enabled unless you know that you need to turn any of them off.
6. Select **Advanced settings**, and accept all the default options if you want to enable S3 Object Lock. This step is optional.
7. Select **Create bucket**.

Step 2: Create Access Keys

Where: AWS Management console

To access Amazon Simple Notification Service (SNS), you must have credentials that Amazon Web Services (AWS) can use to validate your requests. These credentials
must have permissions to access Amazon SNS topics. The following steps provide you details on steps to create access keys using AWS Identity and Access Management (IAM) for security purposes.

1. Log in to AWS Management console and open Amazon Identity and Access Management (IAM) console.
2. On the navigation menu, select Users.
3. Select your user name.
4. In the Security Credentials tab, select Create access key.
5. Copy the Access key ID and Secret access key in the display. Paste them in a clipboard.
6. To download the keys, select Download.csv file icon. This way you can store the file in a secure location.

Step 3: Add an Amazon S3 Cloud Storage Link

Where: Database Actions: Manage Cloud page

Before you create a live table feed, you must establish a connection to the cloud store you want to use.

1. Click the Manage Cloud Store button at the top of the page to go to the Manage Cloud page. For further instructions on adding source files residing in cloud storage provided by Amazon S3, refer to Create an Amazon S3 Cloud Storage Link topic in Managing Cloud Storage Connections.

Note:

Paste the Access key ID and Secret access key generated in the previous step (Step 2: Create Access Keys) to their respective text fields in the Add Cloud Storage page.

Step 4: Create and configure a live table feed to use notifications, and copy the notification URL

Where: Database Actions: Live Feeds page

Creating a live table feed enables you to load data in real time from external storage sources to your table in ADB. External storage you use include as Oracle Object Store, AWS S3 or Microsoft Azure containers.

You can configure a new or an existing live table feed to use notifications:

1. Go to the Database Actions Live Feeds page, as described in Feeding Data.
2. Create or edit a live table feed object, as described in Create a Live Table Feed Object or Edit a Live Table Feed Object. Select the Enable for Notification option
3. Click Create or Save.
4. Click the Actions (three vertical dots) icon on the card for your live feed, and select Show Notification URL.
5. In the **Notification URL** dialog box, click the **Copy** icon to copy the URL to the clipboard. You may want to copy it to a temporary file, so you can retrieve it later. You will use this URL in the subsequent step (Step 7: Create a notifications service subscription).

**Step 5: Create a notifications service subscription topic**

*Where:* Amazon Simple Notification Service (SNS) console

You receive Amazon S3 notifications using Amazon Simple Notification Service (Amazon SNS) topic. You need to add a notification configuration to your bucket using an Amazon SNS topic. SNS topics are shared locations which are used to send notifications of various events that happen in AWS buckets.

During creation, you select a topic name and topic type. After creating a topic, you cannot change the topic type or name. All other configuration choices are optional during topic creation, which you can edit later.

To access any AWS service, you must first create an AWS account.

Navigate to the AWS Management console, and then select **Create an AWS Account**.

Follow the instructions as provided in the Amazon SNS link to create your first IAM administrator user and group. Now you can log in to any of the AWS services as an IAM user.

1. Log in to Amazon SNS console as an IAM user.
2. On the **Topics** page, select **Create topic**.
3. Specify the following fields on the **Create topic** page, in the **Details** section.
   - **Type:** Standard (Standard or FIFO)
   - **Name:** notify-topic. For a FIFO topic, add fifo to the end of the name.
   - **Display Name:** This field is optional.
4. Expand the **Encryption** section and select **Disable encryption**.
5. Expand the **Access policy** section and configure additional access permissions, if required. By default, only the topic owner can publish or subscribe to the topic. This step is optional. Edit the JSON format of the policy based on the topic details you enter. Here is a sample of Access policy in JSON format.

```json
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {"AWS": "*"},
      "Action": [
        "SNS:Publish",
        "SNS:RemovePermission",
        "SNS:SetTopicAttributes",
        "SNS:DeleteTopic",
        "SNS:ListSubscriptionsByTopic",
        "SNS:GetTopicAttributes",
        "SNS:AddPermission",
        "SNS:Subscribe"
      ]
    }
  ]
}
```
6. Expand the **Delivery retry policy (HTTP/S)** section to configure how Amazon SNS retries failed message delivery attempts. This step is optional.

7. Expand the **Delivery status logging** section to configure how Amazon SNS logs the delivery of messages to CloudWatch. This step is optional.

8. Expand **Tags** section to add metadata tags to the topic. This step is optional.

9. Select **Create topic**.

10. The topic’s Name, ARN (Amazon Resource Name), and Topic owner's AWS account ID are displayed in the Details section.

11. Copy the topic ARN to the clipboard.

Step 6: Enable and configure event notifications using the Amazon S3 console

**Where**: Amazon S3 Management console

You can enable Amazon S3 bucket events to send a notification message to a destination whenever those events occur. You configure event notifications for your S3 bucket to notify OCI when there is an update or new data available to load. The following steps explain the procedure to be followed in Amazon S3 console to enable event notifications.
1. Log in to Amazon S3 Management console and sign in as an IAM (Amazon Identity and Access Management) user.

2. In the Buckets list, select the name of the bucket i.e. testbucket. This is the bucket that you had created in Step 1: Create your object store bucket in Amazon S3.

3. Select Properties icon.

4. Navigate to the Event Notifications section and select Create event notification icon.

5. In the General configuration section, specify the following values for event notification.
   - **Event name**: bucket-notification
   - **Prefix**: This value is to filter event notifications by prefix. It is an optional value. This is added to filter event activity.
   - **Suffix**: This value is to filter event notifications by suffix. It is an optional value. This is added to filter event activity.

6. In the Event types section, select one or more event types that you want to receive notifications for. If you are unsure of what event types to pick, then select the All object create events option.

7. In the Destination section, select SNS Topic as the event notification destination.

     **Note:**

     Before you can publish event notifications, you must grant the Amazon S3 the necessary permissions to call the relevant API. This is so that it can publish notifications to a Lambda function or an SNS topic.

8. After you select SNS topic as the event notification destination, select the SNS topic i.e. notify-topic from dropdown. This is the topic you created in Step 5: Create a notifications service subscription topic.

9. Select Save changes.

**Step 7: Create a notifications service subscription**

*Where:* Amazon SNS console

Every Amazon SNS topic has a set of subscriptions. Once a message is published to a topic, SNS handles distributing the message to all its subscribers. The subscribers can be AWS Lambda functions, HTTP(S) endpoints, email addresses and mobile phone numbers capable of receiving SMS messages.

Amazon SNS matches the topic to a list of subscribers who have subscribed to that topic and delivers the message to each of those subscribers.

1. Log in to Amazon SNS console.

2. In the left navigation pane, select Subscriptions.

3. Select Create subscription on the subscriptions page.

4. In the Details section of the Create subscription page, specify the following values.
- **Topic ARN:** Paste the ARN value copied from previous step (Step 5: Create a notifications service subscription topic).
- **Protocol:** HTTPS
- **Endpoint:** Paste the endpoint value you copied while creating the live table feed in previous step (Step 4: Create and configure a live table feed to use notifications and copy the notification URL).

5. Expand the **Subscription filter policy** section to configure a filter policy. This step is optional.

6. Expand the **Redrive policy (dead-letter queue)** section to configure a dead-letter queue for the subscription. This step is optional.

7. Select **Create subscription**.

**Note:**

HTTP(S) endpoints, email addresses, and AWS resources in other AWS accounts require confirmation of the subscription before they can receive messages.

---

**Step 8: Confirm that the endpoint can receive notifications**

Where: Database Actions: Live Feeds page

1. Return to the Database Actions Live Feeds page and find the card for the live table feed you are configuring for a notification-based feed.
2. Click the **Actions** (three vertical dots) icon on the card, and select **Show Confirmation URL**.
3. In the **Confirmation URL** dialog box, click the link to confirm the URL. This does not close this dialog box. If the link is successful, a message is displayed that confirms the subscription is active.
4. Return to the **Confirmation URL** dialog box and select the Check only when the cloud store confirmation process is complete check box, and click **OK**.

Once you finish the above steps, any new files uploaded to the bucket will automatically be loaded into the live table feed table.

For more information on how to enable and configure event notifications using the Amazon S3 console, refer Enabling and configuring event notifications using the Amazon S3 console.

---

**Creating a Notification-Based Live Table Feed using Microsoft Azure**

A notification-based Live Table Feed is an interface between Oracle Cloud Infrastructure and a third-party cloud message queuing service such as Azure Event Grid.

The following section explains the procedure to generate automatic live feed messages using Microsoft (MS) Azure as the cloud storage. When there is an update in the container and the notification conditions are met, a log message is generated and displayed in the live feed in Oracle Cloud Infrastructure.

To create a notification-based live feed with Microsoft Azure as cloud storage you must:

- **Step 1: Create a resource group in Microsoft Azure**
• Step 2: Create a storage account in Microsoft Azure
• Step 3: Create Access Keys
• Step 4: Create a container
• Step 5: Add cloud storage using Microsoft Azure cloud store
• Step 6: Create and configure a live table feed to use notifications and copy the notification URL
• Step 7: Enable Event Resource Provider
• Step 8: Create Event subscription
• Step 9: Confirm that the endpoint can receive notifications

Tip:
To complete the steps above, you will need to alternate between Microsoft Azure portal and Oracle Database Actions pages. You may find it convenient to open the Microsoft Azure portal in one browser page or tab and Database Actions in another, so it is easy to move back and forth.

Step 1: Create a resource group in Microsoft Azure
Where: Microsoft Azure Portal

Resource groups are logical containers where you can manage Azure resources like storage accounts. Resource groups are created so you can deploy, update and delete them as a group. You can create a resource group by following these steps:

1. On the Azure portal, click the **Resource groups** button.
2. Select **Add**.
3. Enter the following values:
   • **Subscription**: Select your Azure subscription, such as Microsoft Azure Enterprise.
   • **Resource group**: Enter a new resource group name, such as resource-group.
   • **Region**: Select your location, such as US west.
4. Click **Review+create**.
5. Click **Create**. It takes a few seconds to create a resource group.

Step 2: Create a storage account in Microsoft Azure
Where: Microsoft Azure Portal

An Azure storage account contains all your storage data objects like blobs, tables, disks etc. You can create a storage account inside the resource group. It provides a unique namespace for your data. To create a storage account, do the following:

1. From the left portal menu, select **Storage accounts** to display a list of your storage accounts.
2. On the Storage accounts page, click the **Create** icon.
3. On the Basic tab, provide the following information for your storage account.
- **Subscription:** Microsoft Azure Enterprise
- **Resource group:** resource-group
- **Storage account name:** teststorage
- **Region:** Select your location, such as US west.
- **Redundancy:** Locally-redundant storage (LRS)

4. You can select **Review+create** to accept the default options and proceed to validate the account.

5. After the validation passes, you can proceed to click on **Create storage** account. In case the validation fails, the portal indicates which settings must be modified.

**Step 3: Create access Keys**

*Where:* Microsoft Azure Portal

You must grant Microsoft Azure the permissions necessary to obtain access keys on your storage locations. The access keys specific to the storage account are generated automatically after the storage account is created in the previous step. The following steps describes the procedure to create access keys.

1. In **Security+Networking**, select **Access keys**. Your account access keys appear with the complete connection string for each key.

2. Select **Show keys** to show your access keys and connection string for each key and to copy values.

3. Copy the connection string value under key1. This value will be pasted in **Azure storage account access key** text field of next step (*Step 5: Add cloud storage using Microsoft Azure cloud store*).

4. Copy the storage account name i.e. teststorage and paste it in **Azure storage account name** text field of next step (*Step 5: Add cloud storage using Microsoft Azure cloud store*).

5. Test the credentials to see if it works or not.

**Step 4: Create a container**

*Where:* Microsoft Azure Portal

A container is a location (also known as buckets in Amazon S3 and OCI) which holds Azure Blob (Binary large object) storage. Follow these steps to create a container.

1. Navigate to your new storage account in the Azure portal.

2. In the left menu for the storage account, scroll to the **Data storage** section, then select **Containers**.

3. Click the **+Container** icon.

4. Enter the name for your new container. The container name must be lowercase, must start with a letter or number, and can include only letters, numbers, and the dash character.

5. Set the level of **Public Access Level** to Private. The default level is Private.

6. Select **Create** to create the container.
Step 5: Add cloud storage using Microsoft Azure cloud store

Where: Database Actions: Manage Cloud page

1. Click the Manage Cloud Store button at the top of the page to go to the Manage Cloud page. For further instructions on adding source files residing in cloud storage provided by Microsoft Azure cloud storage, refer to Create an Microsoft Azure Cloud Storage Link topic in Managing Cloud Storage Connections section.

Note:
Paste the connection string value under key 1 of previous step Step 3: Create Access Keys in Azure storage account access key text field of Add Cloud Storage page. Also paste the storage account name generated in the previous step Step 3: Create Access Keys in Azure storage account name text field of Add Cloud Storage page.

Step 6: Create and configure a live table feed to use notifications and copy the notification URL

Where: Database Actions: Live Feeds page

The Live table feed object enables data to be loaded from Microsoft Azure cloud storage with no polling delay. This object creates an integration between Oracle Cloud Interface and Microsoft Azure.

You can configure a new or an existing live table feed to use notifications:

1. Go to the Database Actions Live Feeds page, as described in Feeding Data.
2. Create or edit a live table feed object, as described in Create a Live Table Feed Object or Edit a Live Table Feed Object. Select the Enable for Notification option
3. Click Create or Save.
4. Click the Actions (three vertical dots) icon on the card for your live feed, and select Show Confirmation URL.
5. In the Notification URL dialog box, click the Copy icon to copy the URL to the clipboard. You may want to copy it to a temporary file, so you can retrieve it later. You will use this URL in the subsequent step, (Step 8: Create Event subscription).

Step 7: Enable Event Resource Provider

Where: Microsoft Azure Portal

If this is the first time you are using Event Grid, you must enable Event Grid resource provider.

1. Select Subscriptions on the left menu.
2. Select the subscription you are using for Event Grid i.e. Microsoft Azure Enterprise.
3. On the left menu, under Settings, select Resource Providers.
5. Select **Register**.

It takes a minute for the registration to finish.

**Step 8: Create Event subscription**

*Where:* Microsoft Azure Portal

You create an Event Subscription by configuring the subscription and specifying the endpoint that will receive the notifications.

1. Select the storage account you created in **Step 2: Create a storage account in Microsoft Azure**.
2. Select the **Events** icon in the left navigation pane.
3. Click on **+Event Subscription**.

The **Create Event Subscription** window appears.

1. Specify the following fields in **Event Subscription** details section:
   - **Name:** Eventssub. This is the name of the Event Subscription we create.
   - **Event Schema:** Event Grid Schema
2. Specify the following fields in **Topic Details** section:
   - **Topic Type:** Storage account
   - **System Topic Name:** eventtopic.
3. Specify the following fields in **Event Types** section:
   - **Event Type:** MicrosoftStorage.BlobCreated
4. Specify the following fields in **Endpoint Details** section:
   - **Endpoint Type:** Web Hook
   - **Endpoint:** Paste the Notification URL copied in **Step 6: Create and configure a live table feed to use notifications and copy the notification URL**.
5. Select **Create**.

This way Microsoft Azure creates a system topic first and then the Event Subscription for the topic.

**Step 9: Confirm that the endpoint can receive notifications**

*Where:* Database Actions: Live Feeds page

1. Return to the Database Actions Live Feeds page and find the card for the live table feed you are configuring for a notification-based feed as created in **Step 6: Create and configure a live table feed to use notifications and copy the notification URL**.
2. Click the **Actions** (three vertical dots) icon on the card, and select **Show Confirmation URL**.
3. In the **Confirmation URL** dialog box, click the link to confirm the URL. This does not close this dialog box. If the link is successful, a message is displayed that confirms the subscription is active.
4. Return to the Confirmation URL dialog box and select the Check only when the cloud store confirmation process is complete check box, and click OK.

Once you finish the above steps, upload a new file to the Microsoft Azure container you created in Step 4: Create a container.

1. Navigate to the container you created.
2. Select the Container to show a list of blobs it contains.
3. Select Upload button to open your local repository and browse the file you need to upload as a block blob.
4. Select the Upload button to upload the blob.
5. You can now view the new blob listed within the container.
6. Return to the Database Actions Live Feeds page and find the card for the live table feed you are configuring for a notification-based feed.
7. Click the Actions (three vertical dots) icon on the card, and select Live table feed Run Details.

You should be able to view logs for the blob uploaded to the Live Feed table from the Microsoft Azure storage in Live table feed Run Details window.

For more details on how to create a topic and subscription in Azure portal, refer to Azure Event Grid Notifications.
The Catalog Page

The Catalog page displays information about entities in the Oracle Autonomous Database. To reach the Catalog page, click Catalog in the Database Actions page, or click the Selector icon and select Catalog from the Data Tools menu in the navigation pane.

The following topics describe the Catalog page and how to use it.

- About the Catalog Page
- Sorting and Filtering Entities
- Searching for Entities
- Viewing Entity Details

About the Catalog Page

Use the Catalog page to get information about the entities in and available to your Oracle Autonomous Database. You can see the data in an entity, the sources of that data, the objects that are derived from the entity, and the impact on derived objects from changes in the sources.

The Catalog page lists and displays details about:

- Database objects such as business models, cloud storage links, and tables that are created by Database Actions applications such as Data Analysis and Data Load.
- Database data dictionary objects such as tables, columns, database links, analytic views, packages, and procedures that have been created by the database tools or a database application such as SQL Developer.

When you first open the Browse catalog page, it contains the Search catalog field, the Catalog User Preferences button, a list of recently viewed objects, and several suggested searches. You can enter a search string, click on one of the recent objects to see its details, or click on Show search suggestions icon to view suggestions on the right side of the page.
As soon as you perform a search or select an item, the page is refreshed with additional items, as described below.

The Catalog page contains:

1. **Search catalog field**
   You can:
   - Click the field and type or paste a new search string into it.
   - Click the field and edit a search string already in the field.
   - Click the field and build a query by selecting items from the drop-down list. You can modify an existing query or build a new one. The query scope icon takes the current search query and turn it to saved query scope.
   The saved query drop-down besides this field lets you search for the entities from the selected query scopes that you had previously saved using the query scope icon.
   Click X to clear your searches in the Catalog field.
   For details about how to construct a search string, see Searching for Entities.

2. **Toolbar**
   The toolbar appears after you run an initial search. It contains these buttons:
• **Sort By**
To select sorting values, click the **Sort By** button to open the list of options. Then click the **Ascending** or **Descending** icon next to one or more of the sorting values. For example, if you select the **Ascending** icon next to **Entity name** and the **Descending** icon next to **Entity type**, the entities will be sorted in alphabetical order by entity name and then in reverse alphabetical order by entity type.

Click **Reset** in the list to clear the choices in the list.

The sorting values you choose are listed next to the **Sort by** label beneath the toolbar. Click the **X** icon on a sorting value to remove it.

• **Page size**
By default, up to 25 entities are displayed on the page. If you want more entities on a page, select a number from this list.

• **Previous** and **Next**
If the search results are displayed on multiple pages, click these buttons to navigate through the pages.

• **Refresh**
Click to refresh the entities shown on the page, based on the current search string.

• **Entity view options**
Choose one of these three options to set how entities are displayed on the page.

Click **Open Card view** to display entities as card arranged into one or two columns; click **Open Grid View** to display entities as rows in a table; or click **Open List View** to display entities in a single column of panels.

The above views display information about the entity, including name, type, owner, application, and other details, depending on the entity view option and on the entity type.

Click **Action** and select **View Details** to view the details of an entity. In the Card view and the List view, you can also click the name of the entity to show its details.

• **Suggestions**
Click to open or close the **Suggestions** panel.

3. **Catalog User Preferences**
When you first open the Catalog, this button appears to the top of the **Search catalog** field. After performing a search, it appears in the toolbar.

Click **Catalog User Preferences** to set the behavior of the Catalog. When you set these options, they take place immediately and are also saved as the default behavior for the page. Clicking Catalog User Preferences opens the General tab where you can view the following options.
• **Show system tables**
  Select this option to include system tables in the search results.

• **Show private tables**
  Select this option to include private tables in the search results.

• **Page size**
  Select the number of entities to display on the page.

• **Entities view**
  Select how entities are shown on the page. There are three options: **Card view** (default) displays entities as cards arranged into one or two columns; **Grid view** displays entities as rows in a table; and **List view** displays entities in a single column of panels.

• **Search criteria**
  Enter a search string that will be used when you click **User configured search** in the **Suggestions** panel on the right side of the Catalog page. For example:

  ```
  owner: your_schema AND (type: TABLE OR type: ANALYTIC_VIEW OR type: VIEW)
  ```

• **Save last 5 search queries**
  Select this option to save the five previous search queries that you entered into the **Search catalog** field. When the **Search catalog** field is empty and you click in the field, the last five search queries are listed in the drop-down list. (Any predefined searches selected from the **Suggestions** panel won't appear in this list, unless you edited them to make a new search.)

  Gradually progress to use the **Query Scopes** tab to view, create, and delete query scopes. You can search for catalogs and save this search using query scopes. See **Searching for Entities** for exploring this tab.

4. **Filters** panel
   Select one or more filter values to limit the entities shown on the page. Only those entities that match the filter values are shown. That is, the items returned by a search are filtered by these filter settings. Selecting all or none of the options shows all entities. See **Sorting and Filtering Entities**.

5. **Sort by** settings
   When you set sorting values by using the **Sort By** control in the toolbar (see above), the settings are displayed in small boxes beneath the toolbar. You can delete a setting by clicking the **X** icon in the box. Or you can change the settings by returning to the **Sort By** control in the toolbar.

6. **Display area**
   The area beneath the **Search catalog** field displays the entities returned by a search and that match the filter criteria set in the **Filters** panel. You can sort the entities by clicking the **Sort By** button and then setting sort values. See **Sorting and Filtering Entities**.
Click the name of the entity or click Action to view details about the entity. See Viewing Entity Details.

7. **Suggestions** panel

The items listed under Suggestions represent predefined queries (search strings) that search for the items described in the link. For example, **All tables** is associated with the search string type:TABLE. When you select one of the suggested search strings, the string is inserted into the Search catalog field, the query is run, and the results are displayed on the page. You can modify the search string in the Search catalog field to modify the search.

---

**Sorting and Filtering Entities**

On the Catalog page, you can sort and filter the displayed entities.

**Sort Entities**

To set a sort order for the entities on the page,

1. Click Sort by on the toolbar.
2. Find the value you want to use to sort, for example Entity Type.
3. Click the Ascending icon or the Descending icon next to that value.

After you've selected a value, it is displayed in a box next to the Sort by label beneath the toolbar.

4. You can set one or more parameters. If you want to add a parameter to your search, repeat the steps above. For example, you can set Entity name (ASC) and then set Created on (DESC). Both appear as separate boxes next to the Sort by label beneath the toolbar.

To remove a sort value, you can do either of the following:

- Click the X icon in the box that shows the sort value (next to the Sort by label beneath the toolbar).
- Click Sort by on the toolbar and click the Ascending or Descending icon next to the value you want to remove.

To return to the default (Updated on DESC), click Sort by and select Reset.

**Filter Entities**

Restrict which entities that are returned by a search are displayed on the page by setting filters in the Filters panel on the left side of the Catalog page. Select one or more filter values. Only those entities that are returned by a search and that match the filter values are shown. Selecting all or none of the options shows all entities returned by the search.
By default, system tables and private tables are not displayed. To display them, click **Show User Preferences** in the toolbar and then click the **Show system tables** slider or the **Show private tables** slider, or both.

---

**Searching for Entities**

Search for entities in the Catalog by entering a search string (query) into the **Search catalog** field at the top of the Catalog page. You can type or paste in a new search string, edit an existing one, or construct a search string by clicking in the field and selecting items from the drop-down list.

The search operates on the entities that meet the filter criteria you have set in **Filters** panel. To clear the **Search catalog** field, click the **X** icon.

**Entering Search Strings**

Syntax and instructions for creating search strings are presented below.

You can enter a search string into the **Search catalog** field in several ways:

- Click an item under **Suggestions** on the right side of the page. When you click one of these items, a search string is entered into the **Search catalog** field and the search is run.

- Type or paste a string directly. When you start typing, suggestions are shown in the drop-down list below the **Search catalog** field. You can keep typing or you can select a suggestion from the list (see next bullet).

- Select or construct a search string by clicking the **Search catalog** field and selecting items.

  If the **Search catalog** field is empty, the drop-down list contains basic search parameters you can select to start building the query. When you select one, or when you click one of the links under **Suggestions**, the string is inserted in the **Search catalog** field, and the drop-down list is updated to present possible additional parameters. You can choose the additional parameters, or edit the string directly, or both.

  If the **Search catalog** field is empty and if you set **Save last 5 search queries** in your preferences, the last five search strings that you entered into the **Search catalog** field will also appear in the drop-down. (Searches initiated by clicking one of the items under **Suggestions** won’t appear in this list.)

When you finish constructing the string, press **Enter** or click the arrow button on the right side of the field.

Click the → icon at the end of the Search field to accept the default query and view all the current owner’s tables, views, and analytic views.

Enter a string in the **Search Catalog** field to find the entities (schema, table, or view) whose label includes the specific string you enter. Click on the **Query Scope** icon to save your search. This allows you to store frequently used searches and access them instantly without any inconvenience.

Selecting Query Scope icon prompts a **Catalog User Preferences** wizard which enables you to save your catalogs based on your preferences.
You can view, create, and edit the previously saved query scopes in this wizard. The query scopes are categorized based on by whom its created.

- Select **Custom** to view, create, edit, or delete the query scopes created by you.
- Select **Predefined** to view the query scopes which are already defined by the Database Actions. This option does not allow you to create, edit, or delete the query scopes.
- Select **All** to view all the query scopes. This includes Custom and Predefined query scopes.

Specify the following fields in the Query Scope tab.

- **Name**: Enter the name of the Query Scope. This is a mandatory field.
- **Label**: This is a mandatory field. Enter a descriptive name here. You will use this field to refer to a query scope.
- **Definition**: Enter the Oracle Autonomous Database Data Definition Language (DDL) that creates the search entity. This is the same search criteria you enter in the Search Catalog field.

Click **Create** to create the Query Scope. Click **Cancel** to cancel its creation.

Once you have created the new query scope, it is visible in the list of query scopes in the Catalog User Preferences wizard.

This wizard also appears on selecting **Catalog User Preferences**. See **Catalog User Preferences** for details.

Create a saved search

You can save your time from redefining the same search again in the future. You can diagnose problems faster since you are just few clicks away from accessing a saved search. Here is how you can create a saved search.

1. From the browse objects drop-down list, select the objects from where you want to search the entities (schema, table, or view).
2. Specify the search criteria in the **Search Catalog** field. For example, you want to search for a specific Entity type and a specific owner.
3. Click **Add** to Saved Searches icon.
4. Clicking the icon prompts the Catalog User Preferences wizard.

5. Enter the name of the saved search in the Title field. The wizard automatically generates the Scope of the search. The definition of the search is also automatically created by concatenating fields used in the Search Catalog field, for example, type: TABLE units AND owner: ADPTEST.

6. Enter description of the search in Description field. This is not a mandatory step.

7. Select Create to create the saved search. Click Cancel to cancel its creation.

After the creation of the saved search, it appears on the list of Saved Searches. You can change the columns displayed in the search results by clicking the pencil icon in the Actions column. Click the delete icon in the Actions column to delete the search you save. The saved searches you create are available for selection in the Saved Search panel in the right of the Catalog page.

Basic Structure

The query string consists of a set of search terms. Here are some examples.

- sales
- type:TABLE
- owner!=SH
- #deployment
- type:TABLE,VIEW

Combine search terms by using the two Boolean operators AND and OR:

- sales and type:TABLE
- sales or type:TABLE

If you don't specify an explicit operator, then AND is assumed. All of the following are equivalent:

- sales type:TABLE owner:sh
- sales AND type:TABLE owner:sh
- sales type:TABLE AND owner:sh
- sales AND type:TABLE AND owner:sh

Negate a single search term by prefacing it by either NOT or by a - (hyphen). All of the following exclude tables from the search:

- sales and NOT type:TABLE
- sales -type:TABLE

Enclose search terms in parentheses to control order:

- sales and (type:TABLE or type:VIEW)
- sales and NOT (type:TABLE or type:VIEW)

If you don't add parentheses, the operators are read from left to right. The following are equivalent:

- sales or type:TABLE and owner:SH
Search Terms

Search terms come in three forms:

- Simple Search Terms
- Property Search Terms
- Classification Search Terms

Simple Search Terms

A simple search term is a simple string, with or without quotes.

- Unquoted string: sales
- Single-quoted string: 'sales'
- Double-quoted string: "sales"

All three types compare the given value, sales in the examples above, to the ENTITY_NAME, but they differ in how the comparison works, as shown in the table below:

<table>
<thead>
<tr>
<th>Search String</th>
<th>Equivalent SQL WHERE Clause</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>sales</td>
<td>WHERE REGEXP_LIKE(entity_name, 'SALES')</td>
<td>A plain regular expression.</td>
</tr>
<tr>
<td>'sales'</td>
<td>WHERE REGEXP_LIKE(entity_name, 'sales')</td>
<td>Similar to the unquoted version, but the search term is not automatically converted to uppercase.</td>
</tr>
<tr>
<td>sales costs</td>
<td>WHERE REGEXP_LIKE(entity_name, 'SALES') AND REGEXP_LIKE(entity_name, 'COSTS')</td>
<td>Treated as two independent search terms combined with the AND operator.</td>
</tr>
<tr>
<td>'sales costs'</td>
<td>WHERE REGEXP_LIKE(entity_name, 'sales costs')</td>
<td>Treated as a single search term. You use single quotes to handle spaces and other special characters, such as colons.</td>
</tr>
<tr>
<td>'don''t'</td>
<td>WHERE REGEXP_LIKE(entity_name, 'don''t')</td>
<td>Escaped single quotes within a search string.</td>
</tr>
<tr>
<td>&quot;sales&quot;</td>
<td>WHERE entity_name = 'sales'</td>
<td>The search string is used as an exact match.</td>
</tr>
</tbody>
</table>

Property Search Terms

A property search term is a combination of three items:

1. The name of a search property (for example, name, type, or owner)
2. A search operator (for example, :, >, or !=)
3. A search value (for example, sales, 'sales', or "sales")

For example:

- name:sales
- owner=SH
- daysSinceCreated>20

You can also specify a comma-delimited list of search values when using the operators :, =, or ~=.

- If the operator is : or =, then the condition works like an IN LIST. For example, name:X,Y is equivalent to entity_name IN ('X', 'Y') and to (entity_name = 'X' OR entity_name = 'Y').
- If the operator is !=, then the condition works like a NOT IN LIST. For example, type!=TABLE,VIEW is equivalent to entity_type NOT IN ('TABLE', 'VIEW') and to (entity_type != 'TABLE' AND entity_type != 'VIEW').

The property name must be one of the following predefined strings. Property names are case sensitive, so TYPE, for example, is not supported.

Some properties apply to all entity types, and some properties apply only to a specific entity type, as shown in the two tables below.

The following table shows those properties that apply to all entity types.

<table>
<thead>
<tr>
<th>Property</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>The name of the entity application as defined by the ALL_LINEAGE_APPLICATIONS view. Examples include DATABASE and INSIGHTS.</td>
</tr>
<tr>
<td>created</td>
<td>The timestamp when the entity was created.</td>
</tr>
<tr>
<td>dateCreated</td>
<td>The date when the entity was created. This is the same as created, but</td>
</tr>
<tr>
<td></td>
<td>truncated to the nearest day.</td>
</tr>
<tr>
<td>dateUpdated</td>
<td>The date when the entity was last updated. This is the same as updated,</td>
</tr>
<tr>
<td></td>
<td>but truncated to the nearest day.</td>
</tr>
<tr>
<td>daysSinceCreated</td>
<td>The number of days since the entity was created. The value is zero if the</td>
</tr>
<tr>
<td></td>
<td>entity was created today.</td>
</tr>
<tr>
<td>daysSinceUpdated</td>
<td>The number of days since the entity was last updated. The value is zero</td>
</tr>
<tr>
<td></td>
<td>if the entity was updated today.</td>
</tr>
<tr>
<td>link</td>
<td>The name of the database link where the entity is defined. This can be</td>
</tr>
<tr>
<td></td>
<td>used to search entities in other, linked, databases.</td>
</tr>
<tr>
<td>local</td>
<td>The value YES if the entity is defined within the database itself;</td>
</tr>
<tr>
<td></td>
<td>otherwise, the value NO. Tables or Insights defined in a schema are</td>
</tr>
<tr>
<td></td>
<td>examples of local entities. Objects in cloud storage, such as CSV or</td>
</tr>
<tr>
<td></td>
<td>Parquet files, are examples of entities that are not local.</td>
</tr>
<tr>
<td>name</td>
<td>The name of the entity.</td>
</tr>
<tr>
<td>namespace</td>
<td>The namespace of the entity as defined by ALL_LINEAGE_NAMESPACES.</td>
</tr>
<tr>
<td>oracleMaintained</td>
<td>The value YES, if the entity is created and maintained by Oracle;</td>
</tr>
<tr>
<td></td>
<td>otherwise, the value NO. The ALL_TABLES view is an example of an Oracle</td>
</tr>
<tr>
<td></td>
<td>maintained entity.</td>
</tr>
<tr>
<td>owner</td>
<td>The owner of the entity.</td>
</tr>
<tr>
<td>parent</td>
<td>The full entity path of the parent entity, if it exists.</td>
</tr>
<tr>
<td>parentName</td>
<td>The name of the parent entity, if it exists.</td>
</tr>
<tr>
<td>parentPath</td>
<td>The full entity path of the parent entity, if it exists.</td>
</tr>
<tr>
<td>parentType</td>
<td>The type of the parent, if it exists.</td>
</tr>
<tr>
<td>Property</td>
<td>Meaning</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>path</td>
<td>The full path of the entity.</td>
</tr>
<tr>
<td>rootName</td>
<td>The name of the outermost containing entity. If the entity has no parent, then the rootName is equal to the entity name. If the entity does have a parent, then the rootName is defined, recursively, as the rootName of the parent entity.</td>
</tr>
<tr>
<td>rootNamespace</td>
<td>The namespace of the outermost containing entity. If the entity has no parent, then the rootNamespace is equal to the entity namespace. If the entity does have a parent, then the rootNamespace is defined, recursively, as the rootNamespace of the parent entity.</td>
</tr>
<tr>
<td>type</td>
<td>The entity type, as defined by ALL_LINEAGE_ENTITYTYPES.</td>
</tr>
<tr>
<td>updated</td>
<td>The timestamp when the entity last updated.</td>
</tr>
</tbody>
</table>

Some searchable properties are specific to certain entity types. The following table shows those entity-type-specific properties. When searching for entities with these properties, it will speed up your search to specify the entity type. For example, instead of just searching on numRows, specify the entity type that has the numRows property:

**TABLE AND numRows > 10**

<table>
<thead>
<tr>
<th>Entity Type</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE</td>
<td>numRows - The number of rows in the table.</td>
</tr>
<tr>
<td></td>
<td>status - The status of the object: VALID or INVALID.</td>
</tr>
<tr>
<td></td>
<td>partitioned - Indicates whether the table is partitioned (YES) or not (NO).</td>
</tr>
<tr>
<td></td>
<td>external - Indicates whether the table is an external table (YES) or not (NO).</td>
</tr>
<tr>
<td></td>
<td>sharded - Indicates whether the object is sharded (Y) or not (N).</td>
</tr>
<tr>
<td>COLUMN</td>
<td>dataType - The data type of the column.</td>
</tr>
<tr>
<td></td>
<td>nullable - Indicates whether a column allows NULLs. The value is N if there is a NOT NULL constraint on the column or if the column is part of a PRIMARY KEY. Otherwise, the value is Y.</td>
</tr>
<tr>
<td>PACKAGE</td>
<td>status - The status of the object: VALID or INVALID.</td>
</tr>
<tr>
<td>PROCEDURE</td>
<td>status - The status of the object: VALID or INVALID.</td>
</tr>
<tr>
<td>ATTRIBUTE_DIMENSION</td>
<td>status - The status of the object: VALID or INVALID.</td>
</tr>
<tr>
<td>HIERARCHY</td>
<td>status - The status of the object: VALID or INVALID.</td>
</tr>
<tr>
<td>ANALYTIC_VIEW</td>
<td>status - The status of the object: VALID or INVALID.</td>
</tr>
<tr>
<td>MEASURE</td>
<td>dataType - Data type of the measure, such as NUMBER.</td>
</tr>
<tr>
<td></td>
<td>nullable - Indicates whether a column allows NULLS. The value is N if there is a NOT NULL constraint on the column or if the column is part of a PRIMARY KEY. Otherwise, the value is Y.</td>
</tr>
<tr>
<td></td>
<td>measureType - Type of the OLAP measure:</td>
</tr>
<tr>
<td></td>
<td>• BASE - Base measures store the data</td>
</tr>
<tr>
<td></td>
<td>• DERIVED - Derived measures calculate the data from base measures; also called calculated measures.</td>
</tr>
<tr>
<td>MINING_MODEL</td>
<td>status - The status of the object: VALID, INVALID, or N/A.</td>
</tr>
</tbody>
</table>
### Entity Type Properties

<table>
<thead>
<tr>
<th>Entity Type</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTION</td>
<td>status - The status of the object: VALID, INVALID, or N/A.</td>
</tr>
<tr>
<td>DIRECTORY</td>
<td>status - The status of the object: VALID, INVALID, or N/A.</td>
</tr>
<tr>
<td>EXTERNAL_LOCATION</td>
<td>fileName - The name of a file in Oracle Directories.</td>
</tr>
<tr>
<td></td>
<td>url - The URI of file a file in object storage.</td>
</tr>
<tr>
<td>LIVE_TABLE_FEED</td>
<td>enableNotifications - Indicates whether a live table feed is enabled for notifications (TRUE) or not (FALSE).</td>
</tr>
</tbody>
</table>

#### Note:

You can find the list of query-type-specific properties by running the following query on The SQL Page:

```sql
select *
from (select entity_type,
            JSON_QUERY(annotation, '$.searchProperties') properties
            from all_lineage_entity_types)
where properties is not null;
```

The operator must be one of the following.

<table>
<thead>
<tr>
<th>Property</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>:</td>
<td>Equal to</td>
</tr>
<tr>
<td>~</td>
<td>Represents REGEXP_LIKE For example, the following two search items are equivalent:</td>
</tr>
<tr>
<td>=</td>
<td>type:TABLE is the same as type=TABLE.</td>
</tr>
<tr>
<td>~</td>
<td>For example, sales name~=sales</td>
</tr>
</tbody>
</table>

The value of the property search term is a string. As with the simple search term, this string can be unquoted, single-quoted, or double-quoted. Unquoted strings are converted to uppercase and quoted strings are left as they are.
### Classification Search Terms

Classifications are metadata about an entity, such as a caption or a description. Classification search terms are similar to property search terms, but they work with entity classifications such as captions or descriptions instead of standard entity properties. The name of the classification must be prefixed by a hash tag (#) and may be unquoted or enclosed in double quotation marks ("). A query converts unquoted classifications to uppercase, which means that the following forms of the \texttt{CAPTION} classification are equivalent.

- \#caption
- \#Caption
- \#CAPTION
- \"CAPTION\"

Classifications are, by their nature, multi-lingual in the sense that the value can vary by National Language Support (NLS) language. By default, the query syntax uses the value of the classification in the current NLS language. You can specify a specific NLS language by adding the name of the language after a forward slash. The language can be quoted, but the query converts it to uppercase in all forms.

- \#caption/French
- \#caption/"ITALIAN"

If the name of the specified language contains a space (for example, "CANADIAN FRENCH"), you must either enclose the name in quotes or replace the space with an underscore character (or both). The following are equivalent:

- \#caption/Canadian_French
- \#caption/"CANADIAN FRENCH"
- \#caption/"Canadian_French"

If you use a classification name on its own as a search term, then the search returns entities that have any non-null value for that classification. If you've defined a classification called \texttt{DEPLOYED}, for example, then you can see all entities with the classification by using a simple term:

- \#deployed

As with other search terms, you can negate it using \texttt{NOT} or - (hyphen). To exclude all entities with the \texttt{DEPLOYED} classification, you can use either of the following searches:
• NOT #DEPLOYED
• -#deployed

You can also use classifications with operator/value pairs, with the same semantics used by property search items; for example:

• #caption~=sales
• #description/Spanish=Ventas

Searching on Multiple Properties

The default for this is ENTITY_NAME, so that if you enter a query string like "sales" it will search for entities with "sales" in the name.

You can also list multiple properties and classifications in this argument; for example, entity_name #caption #description, which will cause it to search for entities with sales in their name or in any CAPTION or DESCRIPTION classifications they may have.

Viewing Entity Details

To view details about an entity, click the Actions icon at the right of the entity entry, then click View Details.

For all entities, the details include Lineage and Impact sections. The inclusion of other details, such as Preview and Statistics, varies by entity type.

Preview

Preview displays the data of the entity. For a table, the Preview displays the columns of the table and the data in those columns. You can sort the data in the column into ascending or descending order by clicking the up or down arrow to the right of the column name.

Describe

For an analytic view, the Describe tab has information about the entity, and displays the hierarchies, levels, level depth, dimension tables, level columns, and number of distinct values for a level.

Lineage

Lineage displays all known information about the upstream dependencies of the entity, and therefore how the entity was created and how it is linked to other entities.

For example, for a table that you created in your database, the lineage is just the table. For a table that you created by loading a CSV file from cloud storage, the lineage includes the ingest directive for the data load and the CSV file that is the source of the data.

Pointing to the name of an item in the lineage displays the table name, the application that created it, the type of entity, the path to it, and the schema it is in.

Arrows point from an entity to the entity that it derives from. For example, for a table created in a data load job, an arrow points from the table to the ingest job and an arrow points from the ingest job to the CSV file. If you point to an arrow, then a Links
Information box appears that shows information about the relationship between the two entities.

To view more details about an item, click the Actions icon for the item, then click **Expand**. For a table, the columns of the table are displayed. Pointing to the name of the table or of a column displays the name, application, type, path, and schema of the table or column. To collapse the display, click the Actions icon, then click **Collapse**.

You can increase or decrease the size of the displayed objects by using the + (plus) and - (minus) keys. You can reposition the objects by grabbing a blank spot in the display and dragging vertically or horizontally.

The lineage for some entities, such as analytic views, is more complex. An analytic view entity deployed for a business model has links to columns in a fact table and to hierarchies. The fact table has links to attribute dimensions and, for a data load job, to an ingest directive for the job. The ingest directive has a link to the source file. The attribute dimensions have links to tables for the dimensions. Those tables have links to ingest directives that have link to source files.

**Impact**

Impact shows all known information about the downstream use of an entity, and therefore how a change in the definition of an entity may affect other entities that depend on it. For example, if a table is used in an analytic view, a change to one of the column definitions in the table may affect the mapping from that column to the analytic view.

**Classifications**

For an analytic view and its attribute dimensions, hierarchies, and measures, the Classifications tab displays classifications and their values. Classifications are metadata that applications can use to present information about analytic views. When creating a business model, you may specify the values for the Caption and Description classifications.

**Optimize**

For an analytic view, the Optimize tab has information about caches created for the analytic view. A cache may exist if the advanced option Enable Autonomous Aggregate Cache was selected for the business model for which the analytic view is deployed.

**Statistics**

Statistics display information about the entity. For example, the statistics for a table include the size of the table and the numbers of rows and columns. They also include the names of the columns, their data types, the number of distinct values and null values, the maximum and minimum values, and other information.

The data is represented in the form of histogram which is column statistic which provides more detailed information about data distribution in a table’s columns.

The histograms in the statistics pane can be representative of the following types:

- **Frequency**: In a frequency histogram, each distinct column value corresponds to a single bucket of the histogram. Since each value has its own dedicated bucket, some buckets may have many values, whereas others have few.
- **Top-frequency**: A top frequency histogram is a variation on a frequency histogram that ignores non-popular values that are statistically insignificant.
- **Height-Balanced**: In this histogram, column values are divided into buckets so that each bucket contains approximately the same number of rows.
• Hybrid: A hybrid histogram combines characteristics of both height-based histograms and frequency histograms. This approach enables the optimizer to obtain better selectivity estimates in some situations.

Data Definition

Data Definition displays the Oracle Autonomous Database DDL that created the entity.
The Data Insights Page

The Data Insights page displays information about patterns and anomalies in the data of entities in your Oracle Autonomous Database.

To reach the Data Insights page, click **Data Insights** in the Database Actions page, or click the Selector icon and select **Data Insights** from the Data Tools menu in the navigation pane.

The following topics describe insights and how to generate and use them.

### About Insights

You can generate insights for a table or for the analytic view deployed for data analysis.

The insights that Data Insights generates for the analytic view of a business model can be more useful than those for a table because of the additional metadata that an analytic view provides.

Insights highlight data points as potentially anomalous if the actual value for a measure when filtering on pairs of analytic view hierarchy values or table column values is considerably higher or lower than the expected value, calculated across all hierarchy or column values. Insights highlight unexpected patterns, which you may want to investigate.

Insights are automatically generated by various analytic functions built into the database. The results of the insight analysis appear as a series of bar charts in the Data Insights dashboard.

Data Insights uses the following steps to generate insights:

1. Finds the values of a measure, for example Sales, across all of the distinct pairs of the hierarchy or column values for the measure. If Sales has the hierarchies or columns Marital Status, Age Band, Income Level, and Gender, then the pairs would be the values of each distinct value of each hierarchy or column paired to each distinct value of each of the other hierarchies or columns. For example, if the values of Marital Status are Married and Single, and the values of Age Band are A, B, and C, then the pairs would be Married and A, Married and B, Married and C, Single and A, Single and B, and Single and C. Each distinct value of Marital Status would also be paired with each distinct value of Income Level and Gender, and so on.

2. Estimates an expected value for the measure for each hierarchy or column pair.

3. Calculates the actual value for the measure for each hierarchy or column pair, for example Marital Status = S, Age Band = C, and then the difference between the actual value and the expected value.

4. Scores all of the differences and selects the largest variations between the actual and expected values to highlight as potential insights.

The resulting insights highlight cases where the measure value is significantly larger or smaller for a given hierarchy or column value pair than expected, for example much higher Sales where Marital Status = S and Age Band = C.
Insights for analytic views tend to use the higher levels of a hierarchy because the differences between the estimated and actual values are generally larger than they are for lower level attributes. For example, the difference in dollars between the estimated and actual sales for the entire USA are generally larger than the difference between the estimated and actual sales for a town with a population under 1000. The difference is calculated in absolute values, not percentages.

Insights for tables categorize columns as dimension columns or measure columns based on their data types and cardinality. A VARCHAR2 column is always categorized as a dimension, but a NUMBER column may be either a dimension or a measure. For example, a NUMBER column for YEAR values that has only 10 distinct values in a table with 1 million rows is assumed to be a dimension.

**Generate Insights and View Reports**

Use these procedures to generate Insights and view reports about them.

**Generate Insights**

To generate insights for a table or business model, do the following:

1. In the **Schema** field, select a schema.
2. In the **Analytic View/Table** field, select an analytic view or a table.
3. In the **Column** field, select a column that contains data about which you want gain insights.
4. Click **Search**.

A confirmation notice announces that the request for insights has been successfully submitted. Dismiss the notice by clicking the Close (X) icon in the notice.

A progress bar indicates that the search is in progress and when it has completed. The insights appear in the Data Insights dashboard as a series of bar charts.

To refresh the display of the insights, click **Refresh**. To have refreshes occur automatically, click **Enable Auto Refresh**.

Click **Recent Searches** to view the list of previous insights search.

Select **View Errors** to see any log of error that occurs while its creation. The results appear in a new browser tab.

**View the Report**

The charts in the Data Insights dashboard show the data that contain anomalous results. The bars in a chart show the actual values. The expected values are indicated by green horizontal lines. The bars that are outlined in black contain the most significant differences between the expected and the actual values.

For example, if the fact table for the insights records values about an insurance program, and the measures of the fact table are AGE_CODE, GENDER_CODE, INCOME_CODE, NUM_INSURED, NUM_UNINSURED, and YEAR, then insights might be generated for the NUM_INSURED measure. In that case, the dashboard would have a series of charts labeled YEAR and INCOME_CODE. Each chart would have a value of the related dimension in the upper left corner. For example, an INCOME_CODE chart that has a related AGE_CODE might have the AGE_CODE value 2 in the upper left corner.
Clicking a chart displays more details about it. At the top of the expanded view of the chart is the dimension name and value and a short textual analysis of notable insights. Below the analysis is the chart showing the values and insights about them.

For example, a chart for INCOME_CODE might have at AGE_CODE = 2 at the top, plus the textual analysis. In the chart, the INCOME_CODE values would be on the x-axis and the NUM_INSURED values would be on the y-axis. Pointing to a bar on the expanded chart displays the actual and the expected NUM_INSURED value for that INCOME_CODE and AGE_CODE.

Click the Back button to return to the Data Insights dashboard.

View Previous Reports

To see the results of a previous search, click the Recent Searches icon at the upper right. In the Recent Searches panel, click anywhere in the box for the insights search that you want to see.

To filter the previous searches, enter a value in the search field at the top of the Recent Searches panel.

To close the Recent Searches panel without selecting a search, click the X at the upper right of the panel.
The Data Analysis Tool

The Data Analysis tool enables you to create Analytic Views with multidimensional metadata. You create Analytic Views on top of a fact table with several dimensions and hierarchies. Analytic views refer to tables in the database and allow users to create hierarchies for dimensions.

Select the Data Analysis card in the Database Actions home page to access this tool. You can also access it by clicking the Selector icon and selecting Data Analysis from the Data Tools menu in the navigation pane.

The Data Analysis page is used to obtain information about the Analytic Views available on your Oracle Autonomous Database.

You can select both hierarchies and measures from Analytic Views. Hierarchies are DB objects that allow users to define relationships between various levels or generations of dimension members. As the name implies, hierarchies organize data using hierarchical relationships. With this tool you can analyze and visualize data in different Points of View (POV). You can export the metadata and visualize it with sophisticated tools like Oracle Analytics Cloud (OAC) and Tableau.

Advantages of Data Analysis tool

With Data Analysis tool you can:

• Visualize, analyze and inspect your data clearly and efficiently with pivot tables
• Calculate total number of errors present in the Analytic View you create and provide solutions to minimize the errors
• Automatically display meaningful insights to help you make better decisions
• Analyze your data across dimensions with support for hierarchical aggregation and drill-down
• Share your Analytic Views with the tool of your choice over various options of raw
data consumption to draw meaningful insights and make them accessible to any
user

By identifying relationships among tables and columns, Analytic Views enable your
system to optimize queries. They also open new avenues for analyzing data. These
avenues include data insights, improved hierarchy navigation, and the addition of
hierarchy-aware calculations.

This tool runs complex and hierarchical SQL queries along with SQL extensions in the
background, which simplifies real-time calculations. It makes complex data more
accessible and easier to understand.

The Data Analysis Page

The following section describes searching and obtaining information about Analytic
Views, creating Analytic Views, inspecting your data, discovering insights and
visualizing data using tools like Oracle Analytics Cloud (OAC), Tableau, and Microsoft
Power BI.

Note:

• OAC has in-built tools to search and utilize Analytic Views.
• We have no direct support for Microsoft Power BI, yet its users can map
  their tool to the AV transparency views to avail some of the benefits of
  Analytic Views.

Read these topics for detailed descriptions of the Data Analysis tool features:
• Searching and obtaining information about Analytic Views
• Creating Analytic Views
• Analyzing data

Searching and obtaining information about Analytic Views

When you first open the Data Analysis page, it displays the list of schemas and
Analytic Views. With Select Schema, you can select a preferred Schema from a list of
schemas available in the drop-down.

The Select Analytic Views drop-down enables you to select an available Analytic
View associated with the schema. When you create an Analytic View, it appears in the
drop-down option with your schema. The Refresh AV icon refreshes the contents of
the selected Analytic View.

The Action icon next to the Refresh AV button enables you to manage Analytic Views.
You can Create Analytic View, Edit Analytic View, Compile Analytic View, Show the
Data Definition Language (DDL) that generates the Analytic View or Delete Analytic
View from the menu.
Obtain information about Analytic Views

By default, Analytic Views are filtered by the current user's schema, as indicated by the schema list below the menu-bar. You can remove the selected schema filter by selecting another user's schema. To search for Analytic Views in other schemas, select one of the schemas from the drop-down.

If there is no Analytic View associated with the schema selected, the tool prompts you to create an Analytic View.

Creating Analytic Views

You can create Analytic Views and view information about them. You can also edit and perform other actions on them.

When you create an Analytic View, you identify a fact table that contains the data to inspect. The Generate Hierarchies and Measures button looks at the contents of that table,
identifies any hierarchies in the fact table, and searches for other tables that may contain related hierarchies.

While creating an Analytic View, you can enable or disable the following advanced options:

• Autonomous Aggregate Cache, which uses the dimensional metadata of the Analytic View to manage a cache and that improves query response time.

• Analytic View Transparency Views, which presents Analytic Views as regular database views and enables you to use your analytic tools of choice while gaining the benefits of Analytic Views.

• Analytic View Base Table Query Transformation, which enables you to use your existing reports and tools without requiring changes to them.

Create Analytic View

To create Analytic View, click Create Analytic View to begin the process.

Click Cancel to cancel the creation of the Analytic View at any time.

Specify Attributes of the Analytic View

On the General tab of the Create Analytic View pane, specify the following:

• The name for the Analytic View
• The fact table for the view
• Advanced options

You can also preview the data of the fact table and see statistics about that data.

In the Name field, specify a name of your choice.

The Schema field has the current user's schema. You can only create an Analytic View in that schema.

In the Fact Table field, expand the drop-down list and click More Sources. The Select Sources dialog box has a list of the available tables and views. Select a table or view from the list.

To filter the list, begin typing characters in the Filter field. As you type, the list changes to show the tables or views that contain the characters. Clear the field to show the complete list again. After you select a table or view, click OK.

To enable or disable the advanced options, on the Create Analytic View pane, click the Show Advanced Options icon at the bottom left. Select or deselect options as desired.

To view the data in the fact table and statistics about the data, click the Preview Data button. In the Preview and Statistics pane, the Preview tab shows the columns of the table and the data in the columns.

The Statistics tab shows the size of the table and the number of rows and columns. The statistics may take a few moments to appear, during which time the message, "No statistics available..." may appear. The statistics include the names of the columns, their data types, the number of distinct values and null values, the maximum and minimum values, and other information. The bar graph displays the top unique column values and the number of their occurrences for the selected column. Point to a bar in the graph to see the number of occurrences of the unique value.
Click **Close** to close the Preview and Statistics pane and return to the Create Analytic View pane.

Click on **Generate Hierarchies and Measures** icon.

The Generating Hierarchies and Measures dialog box displays the progress of searching for dimension tables, analyzing the dimension tables and identifying and creating the data sources, joins, hierarchies, and measures to use. When the process completes, click **Close**.

The **Search for Dimension Tables** check box when selected, enables you to search for dimension tables while generating hierarchies and measures.

After the hierarchies and measures are generated, they are displayed under their respective tabs. Review the hierarchies and measures created for you.

Specify the **Name**, **Fact Table** and select **Advanced Options** in the **General** tab of **Create Analytic View** pane. Click **Create** to generate an Analytic View.

**View Data Sources**

The Data Sources tab displays the sources of the data and the relationships among them. It has a graphical display of the fact table and the related dimension tables. For example, a fact table of health insurance data might have columns for geography identifiers, income codes, and gender codes. The Data Sources tab would display items for the fact table and for the geography, income, and gender dimension tables.

You can add hierarchies from data sources even after generating hierarchies from the existing fact table. You can add one or more hierarchies to your new or existing analytic view. Multiple hierarchies can be defined and used in an analytical view, however only one will be used by default.

Right-click the Data Sources tab and select **Add Hierarchy Sources** or select **Add Hierarchy Sources**.

Selecting **Add Hierarchy Sources** launches an **Add Hierarchy Source** dialog box.
You can view all the fact tables and views associated with the analytic view.

In the filter field, you can either manually look for the source or start typing to search for the fact table or views from the list of available fact tables and views. After typing the full name of the source, the tool automatically matches the fact table or view.

Select **Generate and Add hierarchy from Source** to generate analysis and hierarchies associated with the source data you select.

Select **Find and Add Joins** to link all the data sources with the fact table.

Click **OK** to select the source.

The Generating Hierarchies and Measures dialog box displays the progress of analyzing the dimension tables and creating the hierarchies. When the process completes, click **Close**.

---

**Note:**

When you add a hierarchy from the data source, you see the new hierarchy in the list of hierarchies in the Hierarchies tab. You can navigate between the Data Sources tab, the Hierarchies tab, the Measures tab, the Calculations tab. You can add a hierarchy from a source that is not connected by navigating back to the Data Sources tab.
Select **Remove Hierarchy Source** to remove the hierarchies you create from the data sources. You cannot remove hierarchies generated from the fact table you select from this option.

Expand **Joins** to view the **Hierarchy Source**, **Hierarchy Column** and the **Fact column** mapped with the Analytic View. The **Joins** is visible only when the hierarchy table differs from the fact table.

Expand **Sources** to view the fact table associated with the Analytic View. The data model expands to include the data from the source that you added.

Pointing to an item displays the name, application, type, path and the schema of the table. Click the **Actions** (three vertical dots) icon at the right of the item to display a menu to expand or collapse the view of the table.

An expanded item displays the columns of the table. Pointing to a column displays the name, application, type, path, and schema of the column.
The lines that connect the dimension tables to the fact table indicate the join paths between them. Pointing to a line displays information about the join paths of the links between the tables. If the line connects a table that is collapsed, then the line is dotted. If the line connects two expanded tables, then the line is solid and connects the column in the dimension table to the column in the fact table.

**View and Manage Hierarchies**

The Hierarchies tab displays the hierarchies generated by the Analytic View creation tool. The display includes the name of the hierarchy and the source table.

An analytic view must include at least one hierarchy.

To add a Hierarchy, click **Add Hierarchy**. This results in a display as a list of column in that table. Select a column that operates as the detailed level of the hierarchy and be the join-key to the fact table.
To remove the hierarchy, select the hierarchy you want to remove from the list and click **Remove Hierarchy**.

Select **Move Up** or **Move Down** to position the order of the Hierarchy in the resulting view.

Click **Switch Hierarchy to Measure** to change the hierarchy you select to a measure in the Measures list.

You can also **Add Hierarchy** and **Add Hierarchy From Table** by right-clicking the Hierarchy tab.

If you click on a hierarchy name, a dialog box displays the Hierarchy Name and Source.

To change the source, select a different source from the drop-down list.

Select **Add Level** to add a level to the hierarchy. Click **Remove Level** to remove the selected level from the hierarchy.
To view the data in the fact table and statistics about the data, click the **Preview Data** button. In the Preview and Statistics pane, the Preview tab shows the columns of the table and the data in the columns. The Statistics tab shows the size of the table and the number of rows and columns.

If you click on a particular level in the Hierarchy tab, a dialog box displays its respective Level Name, Level Key, Alternate Level Key, Member Name, Member Caption, Member Description, source, and Sort By drop-down. To change any of the field values, enter the value in the appropriate field.

Member Captions and Member Descriptions generally represent detailed labels for objects. These are typically end-user-friendly names. For example, you can caption a hierarchy representing geography areas named GEOGRAPHY_HIERARCHY as "Geography" and specify its description as "Geographic areas such as cities, states, and countries."

To see the measures for the Analytic View, click **Measures** tab. To immediately create the Analytic View, click **Create**. To cancel the creation, click **Cancel**.

**View and Manage Measures**

The Measures tab displays the measures suggested for the Analytic View. It displays the Measure Name, Column, and operator Expression for each measure.

The measures specify fact data and the calculations or other operations to perform on the data.

To add measures, click **Add Measure**. You can view a new measure at the bottom of the measures list. To remove the measure, select the measure you want to remove from the list and click **Remove Measure**.
To alternatively add a measure from the data source, right-click the Measures tab. This pops up a list of columns that can be used as measures. Select one measure from the list.

You can exclude a column from the measures on right-clicking the Measures tab and selecting Remove Measure.

Click **Switch Measure to Hierarchy** to change the measure you select to hierarchy in the Hierarchies list.
You must specify a measure as the default measure for the analytic view; otherwise, the first measure in the definition is the default. Select Default Measure from the drop-down.

To add a measure, right-click the Measures tab and select Add Measure. To remove a measure, select the particular measure you want to remove, right-click on it and select Remove Measure.

You can select a different column for a measure from the Column drop-down list. You can select a different operator from the Expression drop-down list.

In creating an analytic view, you must specify one or more hierarchies and a fact table that has at least one measure column and a column to join to each of the dimension tables outside of the fact table.

Create new calculated measures

You can add measure calculations to a query of an analytic view.

The measures and hierarchies associated with the analytic views enable us to create new calculated measures.

Calculated measures return values from data stored in one or more measures. You compute these measures at run time.

Note:

You can create the measures without increasing the size of the database since the calculated measures do not store the data. However, they may slow performance. You need to decide which measures to calculate on demand.

The Analytic Views provides easy-to-use templates for creating calculated measures.

Once you create a calculated measure, it appears in the list of measures of the Analytic View. You can create a calculated measure at any time which is available for querying in SQL.

The Data Analysis tool provides easy-to-use templates for creating calculated measures.
Click **Add Calculated Measure** to add calculations to the measures. You can view the new calculation with system generated name in the **Calculations** tab.

Click the newly created calculated measure.

In the **Measure Name** field, enter the name of the calculated measure.

You can select preferred category of calculation from a list of options such as Prior and Future Period, Cumulative Aggregates, Period To Date, Parallel Period, Moving Aggregates, Share, Qualified Data Reference, and Ranking using the **Calculation Category** drop-down.

Your choice of category of calculation dynamically changes the **Calculation Template**.

For more details on how to use Calculation templates, see **Using Calculation Templates**.

Select the **Measure** and **Hierarchy** on which you want to base the calculated measures.

Select **Offset** value by clicking the up or the down arrow. The number specifies the number of members to move either forward or backward from the current member. The ordering of members within a level is dependent on the definition of the attribute dimension used by the hierarchy. The default value is 0 which represents POSITION FROM BEGINNING.

The Expression field lists the expressions which the calculated measure uses.

On the creation of the Analytic view, the calculated measure appears in the navigation tree in the Calculated Measures folder.

Click **Create**. A confirmation dialog box appears that asks for your confirmation. Select **Yes** to proceed with the creation of Analytic View.

After creating the Analytic View, you will view a success message informing you of its creation.

On editing the Analytic View you create, you can view the calculated measure in the navigation tree in the Calculations folder.

Click the **Tour** icon for a guided tour of the worksheet highlighting salient features and providing information if you are new to the interface.

Click the **help** icon to open the contextual or online help for the page you are viewing.

Click **Show DDL** to generate Data Definition Language statements for the analytic view.
Edit Analytic View

You might want to edit an Analytic View to make changes to the data sources, the hierarchies, or the measures.

To edit an Analytic View, click the **Action** icon on the Analytic View item, then click **Edit Analytic View**. On the Edit Analytic View screen, select a tab and make changes as desired.

When you have completed the changes, click **Update**.

Using Calculation Templates

The Data Analysis tool provides templates for all of the calculations typically in demand for business intelligence applications.

The following topics describe the types of calculations available as calculation templates in the tool.

- Cumulative Aggregates
- Prior and Future Period
- Period to Date
- Parallel Period
- Moving Aggregates
- Share
- Rank

Cumulative Aggregates

Cumulative calculations start with the first time period and calculate up to the current member, or start with the last time period and calculate back to the current member.

The tool provides several aggregation methods for cumulative calculations:

- **Cumulative Average**: Calculates a running average across time periods.
- **Cumulative Maximum**: Calculates the maximum value across time periods.
- **Cumulative Minimum**: Calculates the minimum value across time periods.
- **Cumulative Total**: Calculates a running total across time periods.

You can choose the measure, the time dimension, and the hierarchy. For selecting the time range see "Choosing a Range of Time Periods" in *Oracle OLAP User's Guide*.

Cumulative Calculation Example

This template defines a calculated measure using Cumulative Minimum.

Cumulative minimum of **SALES** in the **TIME** dimension and **TIME.CALENDAR** hierarchy within ancestor at level **TIME.CALENDAR_YEAR**, total from **beginning to current member**.

These are the results of a query against the calculated measure, which displays values for the descendants of calendar year 2021. The minimum value for quarters
begins with Q1-21 and ends with Q4-21, and for months begins with Jan-21 and ends with Dec-21.

<table>
<thead>
<tr>
<th>TIME</th>
<th>TIME_LEVEL</th>
<th>SALES</th>
<th>MIN_SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1.21</td>
<td>CALENDAR_QUARTER</td>
<td>32977874</td>
<td>32977874</td>
</tr>
<tr>
<td>Q2.21</td>
<td>CALENDAR_QUARTER</td>
<td>35797921</td>
<td>32977874</td>
</tr>
<tr>
<td>Q3.21</td>
<td>CALENDAR_QUARTER</td>
<td>33526203</td>
<td>32977874</td>
</tr>
<tr>
<td>Q4.21</td>
<td>CALENDAR_QUARTER</td>
<td>41988687</td>
<td>32977874</td>
</tr>
<tr>
<td>JAN-21</td>
<td>MONTH</td>
<td>11477898</td>
<td>11477898</td>
</tr>
<tr>
<td>FEB-21</td>
<td>MONTH</td>
<td>10982016</td>
<td>10982016</td>
</tr>
<tr>
<td>MAR-21</td>
<td>MONTH</td>
<td>10517960</td>
<td>10517960</td>
</tr>
<tr>
<td>APR-21</td>
<td>MONTH</td>
<td>11032057</td>
<td>10517960</td>
</tr>
<tr>
<td>MAY-21</td>
<td>MONTH</td>
<td>11432616</td>
<td>10517960</td>
</tr>
<tr>
<td>JUN-21</td>
<td>MONTH</td>
<td>13333248</td>
<td>10517960</td>
</tr>
<tr>
<td>JUL-21</td>
<td>MONTH</td>
<td>12070352</td>
<td>10517960</td>
</tr>
<tr>
<td>AUG-21</td>
<td>MONTH</td>
<td>11108893</td>
<td>10517960</td>
</tr>
<tr>
<td>SEP-21</td>
<td>MONTH</td>
<td>10346958</td>
<td>10346958</td>
</tr>
<tr>
<td>OCT-21</td>
<td>MONTH</td>
<td>14358605</td>
<td>10346958</td>
</tr>
<tr>
<td>NOV-21</td>
<td>MONTH</td>
<td>12757560</td>
<td>10346958</td>
</tr>
<tr>
<td>DEC-21</td>
<td>MONTH</td>
<td>14872522</td>
<td>10346958</td>
</tr>
</tbody>
</table>

Prior and Future Period

The Data Analysis tool provides several calculations for prior or future time periods.

Here are the calculations used for for prior or future time periods:

- **Prior Period**: Returns the value of a measure at an earlier time period.
- **Difference From Prior Period**: Calculates the difference between values for the current time period and an earlier period.
- **Percent Difference From Prior Period**: Calculates the percent difference between the values for the current time period and an earlier period.
- **Future Period**: Returns the value of a measure at a later time period.
- **Difference From Future Period**: Calculates the difference between the values for the current time period and a later period.
- **Percent Difference From Future Period**: Calculates the percent difference between the values for the current time period and a later period.

When creating a calculation for prior or future time periods, you choose the measure, the time dimension, the hierarchy, and the number of periods from the current period.

Prior Period Example

This template defines a calculated measure using Prior Period:

Prior period for measure **SALES** in **TIME** dimension and **TIME.CALENDAR** hierarchy 1 period ago.

These are the results of a query against the calculated measure. The **PRIOR_PERIOD** column shows the value of Sales for the preceding period at the same level in the Calendar hierarchy.

<table>
<thead>
<tr>
<th>TIME</th>
<th>TIME_LEVEL</th>
<th>SALES</th>
<th>PRIOR_PERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020</td>
<td>CALENDAR_YEAR</td>
<td>136986572</td>
<td>144290686</td>
</tr>
<tr>
<td>2021</td>
<td>CALENDAR_YEAR</td>
<td>140138317</td>
<td>136986572</td>
</tr>
<tr>
<td>Q1.20</td>
<td>CALENDAR_QUARTER</td>
<td>31381338</td>
<td>41988687</td>
</tr>
</tbody>
</table>
Period to Date

Period-to-date functions perform a calculation over time periods with the same parent up to the current period.

These functions calculate period-to-date:

- **Period to Date**: Calculates the values up to the current time period.
- **Period to Date Period Ago**: Calculates the data values up to a prior time period.
- **Difference From Period to Date Period Ago**: Calculates the difference in data values up to the current time period compared to the same calculation up to a prior period.
- **Percent Difference From Period To Date Period Ago**: Calculates the percent difference in data values up to the current time period compared to the same calculation up to a prior period.

When creating a period-to-date calculation, you can choose from these aggregation methods:

- Sum
- Average
- Maximum
- Minimum

You also choose the measure, the time dimension, and the hierarchy.

**Period to Date Example**

This template defines a calculated measure using Period to Date.

Gregorian Year to date for SALES in the TIME dimension and TIME.CALENDAR hierarchy. Aggregate using MINIMUM from the beginning of the period.

These are the results of a query against the calculated measure. The MIN_TO_DATE column displays the current minimum SALES value within the current level and year.

<table>
<thead>
<tr>
<th>TIME</th>
<th>TIME_LEVEL</th>
<th>SALES</th>
<th>MIN_TO_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1.21</td>
<td>CALENDAR_QUARTER</td>
<td>36154815</td>
<td>36154815</td>
</tr>
<tr>
<td>Q2.21</td>
<td>CALENDAR_QUARTER</td>
<td>36815657</td>
<td>36154815</td>
</tr>
<tr>
<td>Q3.21</td>
<td>CALENDAR_QUARTER</td>
<td>32318935</td>
<td>32318935</td>
</tr>
<tr>
<td>Q4.21</td>
<td>CALENDAR_QUARTER</td>
<td>34848911</td>
<td>32318935</td>
</tr>
<tr>
<td>JAN-21</td>
<td>MONTH</td>
<td>13119235</td>
<td>13119235</td>
</tr>
<tr>
<td>FEB-21</td>
<td>MONTH</td>
<td>11441738</td>
<td>11441738</td>
</tr>
<tr>
<td>MAR-21</td>
<td>MONTH</td>
<td>11593842</td>
<td>11441738</td>
</tr>
<tr>
<td>APR-21</td>
<td>MONTH</td>
<td>11356940</td>
<td>11356940</td>
</tr>
<tr>
<td>MAY-21</td>
<td>MONTH</td>
<td>13820218</td>
<td>11356940</td>
</tr>
<tr>
<td>JUN-21</td>
<td>MONTH</td>
<td>11638499</td>
<td>11356940</td>
</tr>
<tr>
<td>JUL-21</td>
<td>MONTH</td>
<td>9417316</td>
<td>9417316</td>
</tr>
</tbody>
</table>
Parallel Period

Parallel periods are at the same level as the current time period, but have different parents in an earlier period. For example, you may want to compare current sales with sales for the prior year at the quarter and month levels.

The Data Analysis tool provides several functions for parallel periods:

- **Parallel Period**: Calculates the value of the parallel period.
- **Difference From Parallel Period**: Calculates the difference in values between the current period and the parallel period.
- **Percent Difference From Parallel Period**: Calculates the percent difference in values between the current period and the parallel period.

To identify the parallel period, you specify a level and the number of periods before the current period. You can also decide what happens when two periods do not exactly match, such as comparing daily sales for February (28 days) with January (31 days).

You also choose the measure, the time dimension, and the hierarchy.

**Parallel Period Example**

This template defines a calculated measure using Parallel Period.

Parallel period for **SALES** in the **TIME** dimension and **TIME.CALENDAR** hierarchy 1 **TIME.CALENDAR.QUARTER** ago based on position from **beginning to ending** of period.

These are the results of a query against the calculated measure, which lists the months for two calendar quarters. The parallel month has the same position within the previous quarter. The prior period for **JUL-21** is **APR-21**, for **AUG-21** is **MAY-21**, and for **SEP-21** is **JUN-21**.

<table>
<thead>
<tr>
<th>TIME</th>
<th>PARENT</th>
<th>SALES</th>
<th>LAST_QTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>APR-21</td>
<td>CY2006.Q2</td>
<td>11356940</td>
<td>13119235</td>
</tr>
<tr>
<td>MAY-21</td>
<td>CY2006.Q2</td>
<td>13820218</td>
<td>11441738</td>
</tr>
<tr>
<td>JUN-21</td>
<td>CY2006.Q2</td>
<td>11638499</td>
<td>11593842</td>
</tr>
<tr>
<td>JUL-21</td>
<td>CY2006.Q3</td>
<td>9417316</td>
<td>11356940</td>
</tr>
<tr>
<td>AUG-21</td>
<td>CY2006.Q3</td>
<td>11596052</td>
<td>13820218</td>
</tr>
<tr>
<td>SEP-21</td>
<td>CY2006.Q3</td>
<td>11305567</td>
<td>11638499</td>
</tr>
</tbody>
</table>

Moving Aggregates

Moving aggregates are performed over the time periods surrounding the current period.

The Data Analysis tool provides several aggregation methods for moving calculations:

- **Moving Average**: Calculates the average value for a measure over a fixed number of time periods.
- **Moving Maximum**: Calculates the maximum value for a measure over a fixed number of time periods.
• **Moving Minimum**: Calculates the minimum value for a measure over a fixed number of time periods.

• **Moving Total**: Returns the total value for a measure over a fixed number of time periods.

You can choose the measure, the time dimension, and the hierarchy. You can also select the range, as described in "Choosing a range of time periods" in *Oracle OLAP User's Guide*, and the number of time periods before and after the current period to include in the calculation.

**Moving Aggregates Example**

This template defines a calculated measure using Moving Minimum.

Moving minimum of **SALES** in the **TIME** dimension and **TIME.CALENDAR** hierarchy. Include 1 preceding and 1 following members within **level**.

These are the results of a query against the calculated measure, which displays values for the descendants of calendar year 2021. Each value of Minimum Sales is the smallest among the current value and the values immediately before and after it. The calculation is performed over all members of a level in the cube.

<table>
<thead>
<tr>
<th>TIME</th>
<th>TIME_LEVEL</th>
<th>SALES</th>
<th>MIN_SALES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1.21</td>
<td>CALENDAR_QUARTER</td>
<td>32977874</td>
<td>32977874</td>
</tr>
<tr>
<td>Q2.21</td>
<td>CALENDAR_QUARTER</td>
<td>35797921</td>
<td>32977874</td>
</tr>
<tr>
<td>Q3.21</td>
<td>CALENDAR_QUARTER</td>
<td>33526203</td>
<td>33526203</td>
</tr>
<tr>
<td>Q4.21</td>
<td>CALENDAR_QUARTER</td>
<td>41988687</td>
<td>31381338</td>
</tr>
<tr>
<td>JAN-21</td>
<td>MONTH</td>
<td>11477898</td>
<td>10982016</td>
</tr>
<tr>
<td>FEB-21</td>
<td>MONTH</td>
<td>10982016</td>
<td>10517960</td>
</tr>
<tr>
<td>MAR-21</td>
<td>MONTH</td>
<td>10517960</td>
<td>10517960</td>
</tr>
<tr>
<td>APR-21</td>
<td>MONTH</td>
<td>11032057</td>
<td>10517960</td>
</tr>
<tr>
<td>MAY-21</td>
<td>MONTH</td>
<td>11432616</td>
<td>11032057</td>
</tr>
<tr>
<td>JUN-21</td>
<td>MONTH</td>
<td>13333248</td>
<td>11432616</td>
</tr>
<tr>
<td>JUL-21</td>
<td>MONTH</td>
<td>12070352</td>
<td>11108893</td>
</tr>
<tr>
<td>AUG-21</td>
<td>MONTH</td>
<td>11108893</td>
<td>10346958</td>
</tr>
<tr>
<td>SEP-21</td>
<td>MONTH</td>
<td>10346958</td>
<td>10346958</td>
</tr>
<tr>
<td>OCT-21</td>
<td>MONTH</td>
<td>14358605</td>
<td>10346958</td>
</tr>
<tr>
<td>NOV-21</td>
<td>MONTH</td>
<td>12757560</td>
<td>12757560</td>
</tr>
<tr>
<td>DEC-21</td>
<td>MONTH</td>
<td>14872522</td>
<td>12093518</td>
</tr>
</tbody>
</table>

**Share**

Share calculates the ratio of a measure's value for the current dimension member to the value for a related member of the same dimension.

You can choose whether the related member is:

• **Top of hierarchy**: Calculates the ratio of each member to the total.

• **Member's parent**: Calculates the ratio of each member to its parent.

• **Member's ancestor at level**: Calculates the ratio of each member to its ancestor, that is, a member at a specified level higher in the hierarchy.

When creating a share calculation, you can choose the measure, dimension, and hierarchy. You also have the option of multiplying the results by 100 to get percentages instead of fractions.
Share Example

This template defines a calculated measure using SHARE:

Share of measure SALES in PRODUCT.PRIMARY hierarchy of the PRODUCT dimension as a ratio of top of hierarchy.

These are the results of a query against the calculated measure. The TOTAL_SHARE column displays the percent share of the total for the selected products.

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>PROD_LEVEL</th>
<th>SALES</th>
<th>TOTAL_SHARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Product</td>
<td>TOTAL</td>
<td>144290686</td>
<td>100</td>
</tr>
<tr>
<td>Hardware</td>
<td>CLASS</td>
<td>130145388</td>
<td>90</td>
</tr>
<tr>
<td>Desktop PCs</td>
<td>FAMILY</td>
<td>78770152</td>
<td>55</td>
</tr>
<tr>
<td>Portable PCs</td>
<td>FAMILY</td>
<td>19066575</td>
<td>13</td>
</tr>
<tr>
<td>CD/DVD</td>
<td>FAMILY</td>
<td>16559860</td>
<td>11</td>
</tr>
<tr>
<td>Software/Other</td>
<td>CLASS</td>
<td>14145298</td>
<td>10</td>
</tr>
<tr>
<td>Accessories</td>
<td>FAMILY</td>
<td>6475353</td>
<td>4</td>
</tr>
<tr>
<td>Operating Systems</td>
<td>FAMILY</td>
<td>5738775</td>
<td>4</td>
</tr>
<tr>
<td>Memory</td>
<td>FAMILY</td>
<td>5430466</td>
<td>4</td>
</tr>
<tr>
<td>Modems/Fax</td>
<td>FAMILY</td>
<td>5844185</td>
<td>4</td>
</tr>
<tr>
<td>Monitors</td>
<td>FAMILY</td>
<td>4474150</td>
<td>3</td>
</tr>
<tr>
<td>Documentation</td>
<td>FAMILY</td>
<td>1931170</td>
<td>1</td>
</tr>
</tbody>
</table>

Rank

Rank orders the values of a dimension based on the values of the selected measure. When defining a rank calculation, you choose the dimension, a hierarchy, and the measure.

You can choose a method for handling identical values:

- **Rank**: Assigns the same rank to identical values, so there may be fewer ranks than there are members. For example, it may return 1, 2, 3, 3, 4 for a series of five dimension members.

- **Dense Rank**: Assigns the same minimum rank to identical values. For example, it may return 1, 2, 3, 3, 5 for a series of five dimension members.

- **Average Rank**: Assigns the same average rank to identical values. For example, it may return 1, 2, 3.5, 3.5, 5 for a series of five dimension members.

You can also choose the group in which the dimension members are ranked:

- **Member's level**: Ranks members at the same level.

- **Member's parent**: Ranks members with the same parent.

- **Member's ancestor at level**: Ranks members with the same ancestor at a specified level higher in the hierarchy.

Rank Example

This template defines a calculated measure using Rank:

Rank members of the PRODUCT dimension and PRODUCT.PRIMARY hierarchy based on measure SALES. Calculate rank using RANK method with member's parent in order lowest to highest. Rank NA (null) values nulls last.

These are the results of a query against the calculated measure in which the products are ordered by RANK:
### Analyzing data

The output display pane for an Analytic View enables you to Analyze the data from Analytic View.

There are three tabs displayed in the output display pane for an Analytic View. They are the following:

- **Analyze** tab- In this tab, you select **Hierarchy Levels** and **Measures** from the Analytic View to analyze the data.
- **Data Quality** tab- This tab displays structural errors in the table data and dimension sources for the Analytic View.
- **Export** tab- This tab exports the Analytic View you create to different tools available for better visualization.

#### Analyze tab

The Analytic View Browser displays the Hierarchies, Levels and calculated measures generated with the selected Analytic View. You can drag and drop Levels and Measures from the Analytic View browser to the drop area in Columns, Rows or Values and Filters in the output pane of Analyze tab. The Rows and Columns are interchangeable. The values should be Analytic View measures.

Drag and drop measures to the filter area in the **Analyze** tab. After you drop the selected measure to the drop area, a faceted search window appears.
You can analyze filtered and refined searches in the output by dragging and dropping multiple measures in the filter area. It helps you drill down to the values you are looking for.

Choose one of the two options in the top-right of the dialog box to set the mode of display on the window.

Click **List View** to display the filter condition in two columns. The left column is the search column in which you can drill down to the specific value in the measures. You can select multiple values from the complete list of values displayed in the left column. The right column shows the values you have selected from the list.

Click **Multi-Select** view to display the search condition as a single column. Click **Add Filter** to view the complete list of values. You can select multiple values from this list.

Click the **display or hide funnel** icon to open faceted search list.

The Faceted search list consists of measures and the count next to each value displays the total number of items that fit the category. You can click on attributes of the measures that apply as a filter to your search that further narrows the search down.

Choose one of the three options in the top-right of the dialog box to view the result of the query in the appropriate mode. The modes of display are Table view, Pivot view and Chart view.

Click the **SQL** icon on the display area to view the SQL query of the filter condition. The worksheet editor page appears.
Click the **Run Query** icon to execute the statements in the worksheet editor. The area beneath the SQL worksheet editor displays the results of the search condition that match the filter criteria.

After creating an Analytic View, you can use it to analyze its data in different formats. You can choose from different visualization formats such as pivot, table or chart. These are generated based on the levels and measures you select. You can analyze the data in pivot, table and chart forms by selecting their respective icons displayed in the right corner. These are generated based on the levels and measures you select.

You can add totals to the pivot table data. This **Total** value is created as an additional row and an additional column. The tool automatically calculates the aggregate value based on the values, rows, and columns you drop from the Analytic View browser to the drop area.

The grand total values are displayed at the end of the total row and total column. Based on the grouping of data, the grand total values are further broken down into sub-total values.

This helps in comparing values and summarizing data which further helps to gain insights from the different sets of data you select.

The **Related Insights** panel displays the generated graphically based on measures selected to visualize the data. It is designed in such a way that it could trigger immediate insights.
Data Quality tab

This tab helps you measure the total number of structural errors in the Analytic View you create. For example, null values in a join column is an error. In addition to being a join column, this error will be signaled if there is a null value in the detail level of a hierarchy. This way you can resolve the errors and enhance the quality of the data. This tab displays the fact table and dimensions along with the associated number of errors. Click the triangular icon on the box with errors to expand the list to display first five rows with that specific error as a sample. The errors are categorized based on dimensions and fact tables.

Click the Download icon to view all errors as a CSV file.

Select the cards to view which category the errors belong to. Select Analyze icon to switch the view back to Analyze tab.

Export tab

This tab exports the data to tools like Tableau.

Note:

- OAC has in-built tools to search and utilize Analytic Views.
- We have no direct support for Microsoft Power BI, yet its users can map their tool to the AV transparency views to avail some of the benefits of Analytic Views.

This allows anyone to connect to data, visualize and create interactive dashboards in a few clicks.
Part III
Managing and Monitoring Autonomous Database

This part provides information on managing Autonomous Database.

Topics

• Controlling Your Autonomous Database Instance
• Managing Users on Autonomous Database
• Managing and Monitoring Performance of Autonomous Database
• Backing Up and Restoring Autonomous Database
• Cloning and Moving an Autonomous Database
• Wallet Security, Using Oracle Data Safe, Oracle Database Vault, and Changing License Type
• Managing Encryption Keys on Autonomous Database
• Configuring Network Access with Access Control Rules (ACLs) and Private Endpoints
• Managing and Viewing Maintenance Windows, Patching Options, Time Zone Data Updates, Work Requests, Events and Notifications, and Customer Contacts
• Using Standby Databases with Autonomous Database for Disaster Recovery
• Using Refreshable Clones with Autonomous Database
• Auditing Autonomous Database
Controlling Your Autonomous Database Instance

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

Topics

- Provision Autonomous Database
- Start Autonomous Database
- Stop Autonomous Database
- Restart Autonomous Database
- Rename Autonomous Database
- Update the Display Name for an Autonomous Database Instance
- Terminate an Autonomous Database Instance
- Add CPU or Storage Resources or Enable Auto Scaling
- Remove CPU or Storage Resources or Disable Auto Scaling
- Use Auto Scaling
- Schedule Start and Stop Times for an Autonomous Database Instance
- Change Autonomous Database Operation Mode to Read/Write Read-Only or Restricted
- Use Operations Insights on Autonomous Database
- Use Oracle Data Safe to Assess and Protect Data with Autonomous Database
- Concurrent Operations on Autonomous Database

Provision Autonomous Database

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

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- Choose your region. See Switching Regions for information on switching regions and working in multiple regions.
Choose your **Compartment**. See **Compartments** for information on using and managing compartments.

On the Autonomous Databases page, perform the following steps:

1. Click **Create Autonomous Database**.
2. Provide basic information for the Autonomous Database.
   - **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.
     The default display name is a generated 16-character string that matches the default database name.
   - **Database name**: Specify the database name; it must consist of letters and numbers only. The maximum length is 30 characters. The same database name cannot be used for multiple Autonomous Databases in the same tenancy in the same region.
     The default database name is a generated 16-character string that matches the default display name.
3. Choose a workload type. Select the workload type for your database from the choices:
   - **Data Warehouse**
   - **Transaction Processing**
   - **JSON Database**
   - **APEX**
4. 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.
5. Configure the database.
   - **Always Free**: Select to show Always Free configuration options.
     Always Free does not show when you select **JSON** workload type.
   - **Choose database version**: Select the database version. The available database version is 19c.
     With Always Free selected, the available database versions are: 19c and 21c.
   - **OCPUs count**: Specify the number of CPU cores for your database.
     Your license type determines the **OCPUs count** maximum. For example, if your license type is Bring Your Own License (BYOL) with Oracle Database Standard Edition (SE), the **OCPUs count** maximum is 8.
• **OCPU auto scaling** By default OCPU auto scaling is enabled to allow the system to automatically use up to three times more CPU and IO resources to meet workload demand. If you do not want to use OCPU auto scaling then deselect this option.

See Use Auto Scaling for more information.

• **Storage (TB)** Specify the storage you wish to make available to your database, in terabytes.

• **Storage auto scaling** By default storage auto scaling is disabled. Select if you want to enable storage auto scaling to allow the system to automatically expand to use up to three times more storage.

See Use Auto Scaling for more information.

6. Create administrator credentials. Set the password for the Autonomous Database Admin user.

• **Username** This is a read only field.

• **Password** Set the password for the Autonomous Database Admin user.

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

The password must meet the strong password complexity criteria based on Oracle Cloud security standards. For more information on the password complexity rules, see About User Passwords on Autonomous Database.

7. Choose network access

![Note]

After you provision your Autonomous Database you can change the network access option you select for the instance.

• **Secure access from everywhere** By default, secure connections are allowed from everywhere.

• **Secure access from allowed IPs and VCNs only** This option restricts connections to the database according to the access control lists (ACLs) you specify. To add multiple ACLs for the Autonomous Database, click + Access Control Rule.

See Configure Access Control Lists When You Provision or Clone an Instance for more information.

• **Private endpoint access only** This option assigns a private endpoint, private IP, and hostname to your database. Specifying this option allows traffic only from the VCN you specify; access to the database from all public IPs or VCNs is blocked. This allows you to define security rules, ingress/egress, at the Network Security Group (NSG) level and to control traffic to your Autonomous Database.

See Configure Private Endpoints When You Provision or Clone an Instance for more information.

8. Choose license and Oracle Database Edition

• **Bring Your Own License (BYOL)**
Select if your organization already owns Oracle Database software licenses. Bring your existing database software licenses to the database cloud service. See Cloud pricing for information on Bring Your Own License (BYOL) and other licensing options for Oracle Cloud Infrastructure cloud service pricing.

- Choose Oracle Database Edition
  When you select Bring Your Own License (BYOL), you also choose an Oracle Database Edition. The Oracle Database Edition you select is based on the license you bring to Autonomous Database and changes the maximum value that you can select for the OCPU count. The choices are:

  **Oracle Database Enterprise Edition (EE):** For this license type the maximum allowed value for OCPU count is 128, however you may contact your Oracle account team to request more than 128 OCPUs. With auto scaling enabled you can use up to OCPU count x 3 OCPUs. For example, if you set the OCPU count to 128, you can use up to 384 OCPUs.

  **Oracle Database Standard Edition (SE):** For this license type the maximum allowed value for OCPU count is 8. With auto scaling enabled you can use up to OCPU count x 3 OCPUs. This license restricts the number of OCPUs you can use to a maximum of 8 OCPUs, with or without auto scaling enabled.

  See Use Auto Scaling for more information.

- License Included
  Subscribe to new database software licenses and the database cloud service.

9. (Optional) Provide up to 10 maintenance contacts
   Click Add Contact and in the Contact Email field, enter a valid email address. To enter multiple Contact Email addresses, repeat the process to add up to 10 customer contact emails.

   See View and Manage Customer Contacts for Operational Issues and Announcements for more information.

10. (Optional) Click Show Advanced Options to select advanced options.

    - Encryption Key
      **Encryption using Oracle-managed keys:** By default Autonomous Database uses Oracle-managed encryption keys. Using Oracle-managed keys, Autonomous Database creates and manages the encryption keys that protect your data and Oracle handles rotation of the TDE master key.

      **Encrypt using customer-managed keys:** If you select customer-managed keys, a master encryption key in the Oracle Cloud Infrastructure Vault is used to generate the TDE master key on Autonomous Database.

      See Use Customer-Managed Encryption Keys on Autonomous Database for more information.

    - Maintenance
      **Patch level** By default the patch level is Regular. Select Early to configure the instance with the early patch level. Note: you cannot change the patch level after you provision an instance.

      See Set the Patch Level for more information.

    - Management
      Choose a character set and a national character set for your database.

      See Choose a Character Set for Autonomous Database for more information.
• **Tags**
  If you want to use Tags, enter the **TAG KEY** and **VALUE**. Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values which can be attached to resources.
  See [Tagging Overview](#) for more information.

11. Click **Create Autonomous Database**.
    On the Oracle Cloud Infrastructure console the Lifecycle State shows **Provisioning** until the new database is available.

---

### Start Autonomous Database

Describes the steps to start an Autonomous Database instance.

Perform the following prerequisite steps as necessary:

1. Open the Oracle Cloud Infrastructure Console by clicking the **next to Oracle Cloud**.
2. From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
3. On the Autonomous Databases page select an Autonomous Database from the links under the **Display Name** column.
4. On the **Details** page, from the **More Actions** drop-down list, select **Start**.
   **Start** is only shown for a stopped instance.
5. Click **Start** to confirm.

**Note:**
When an Autonomous Database instance is started, Autonomous Database CPU billing is initiated, billed by the second with a minimum usage period of one minute.

### Stop Autonomous Database

Describes the steps to stop an Autonomous Database instance.

Perform the following prerequisite steps as necessary:

1. Open the Oracle Cloud Infrastructure Console by clicking the **next to Oracle Cloud**.
2. From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
3. On the Autonomous Databases page select an Autonomous Database from the links under the **Display Name** column.
4. On the **Details** page, from the **More Actions** drop-down list, select **Stop**.
5. Click **Stop** to confirm.
When concurrent operations such as scaling or creating a manual backup are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

**Note:**

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

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

### Restart Autonomous Database

Describes the steps to restart an Autonomous Database instance.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the Display Name column.

1. On the Details page, from the More Actions drop-down list, select Restart.
2. In the confirmation dialog, select Restart to confirm.

When concurrent operations such as scaling or creating a manual backup are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

The Autonomous Database instance state changes to "Restarting". After the restart is successful, Autonomous Database instance state is "Available".

**Note:**

When an Autonomous Database instance is restarted, Autonomous Database CPU billing is initiated, billed by the second with a minimum usage period of one minute.
Rename Autonomous Database

Shows you the steps to change the database name for an Autonomous Database instance.

Before you rename your database, note the following:

- A single tenancy in the same region cannot contain two Autonomous Databases with the same name.
- The database rename operation changes the connection strings required to connect to the database. Thus, after you rename a database you must download a new wallet for any existing instance wallets that you use to connect to the database (and any applications also must be updated to use the new connection string and the new wallet to connect to the database).

If you are using a regional wallet, you can continue to use the existing wallet to connect to any databases that were not renamed. However, if you want to connect to a renamed database, you must download a new regional wallet.

See Download Client Credentials (Wallets) for more information.

Note:
The rename operation terminates all connections to the database. After the rename operation completes, you can reconnect to the database.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the Display Name column.

To rename an Autonomous Database instance, do the following:

1. On the Details page, from the More Actions drop-down list, select Rename Database.
   This shows the Rename Database dialog.
2. On the Rename Database dialog, provide values for the fields:
   - New database name: the name you enter must consist of letters and numbers only. The maximum length is 30 characters.
   - Enter the current database name to confirm the name change

3. Click **Rename Database**.
   After validating the fields, while the system renames the database the lifecycle state changes to **Updating**. You can start using the database after the rename operation completes and the lifecycle state shows **Available**.

Notes for Autonomous Database rename:

- Renaming your database does not change global references to your database from existing database links on remote databases. Changing such references is the responsibility of the administrator of the remote databases.
- When you rename a database, the Autonomous Database OCID does not change.
- If your database has Autonomous Data Guard enabled, the rename operation is not available.
- You cannot use the rename operation on a refreshable clone instance or on a database that is the source for a refreshable clone.
- The rename operation restarts the database.
- If you configured your Autonomous Database for manual backups prior to the rename operation, then your existing manual backups continue to work after the rename operation. If you perform new manual backups after the rename, they are stored in the same bucket that you configured prior to the rename operation.

See [Configure Manual Backups on Autonomous Database](#) for information on manual backups.
• You can rename your Autonomous Database using the API. See `UpdateAutonomousDatabase` for more information.

Update the Display Name for an Autonomous Database Instance

Shows you the steps to change the display name for an Autonomous Database instance.

Perform the following prerequisite steps as necessary:

• Open the Oracle Cloud Infrastructure Console by clicking the ☐ next to Oracle Cloud.
• From the Oracle Cloud Infrastructure left navigation menu click `Oracle Database` and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
• On the Autonomous Databases page select an Autonomous Database from the links under the **Display Name** column.

To update the display name on an Autonomous Database instance:

1. On the **Details** page, from the **More Actions** drop-down list, select **Update Display Name**.

   **Update Display Name**

   ![Update Display Name](image)

   - New display name
   - Enter the current display name `DOC_SALES_TEST` to confirm the name change

   ![Update Display Name](image)

   2. Enter a display name in the **New display name** field.
      • The display name minimum length is 1 character and maximum length is 255 characters.
      • The display name must include only letters, numbers, underscores `_`, and hyphens `-`.
      • The display name must start with a letter or an underscore `_`, and cannot contain two successive hyphens `-`.

   3. Enter the current display name to confirm the change.

   4. Click **Update Display Name**.

   You can see the updated display name on the Oracle Cloud Infrastructure Console.
Terminate an Autonomous Database Instance

Describes the steps to terminate an Autonomous Database instance.

**Note:**
Terminating Autonomous Database permanently deletes the instance and removes all automatic backups. You cannot recover a terminated database.

Perform the following prerequisite steps as necessary:

1. Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
2. From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
3. On the Autonomous Databases page select an Autonomous Database from the links under the Display Name column.
4. On the Details page, from the More Actions drop-down list, select Terminate.
5. On the Terminate Autonomous Database page enter the database name to confirm that you want to terminate the database.

When concurrent operations such as scaling or creating a manual backup are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

6. Click Terminate Autonomous Database.

**Note:**
There are limitations for terminating a database when Autonomous Data Guard is enabled with a cross-region standby database. See Terminate a Cross-Region (Remote) Standby Database for more information.

Add CPU or Storage Resources or Enable Auto Scaling

Describes how to scale your Autonomous Database on demand by adding CPU cores or storage. Also describes how to enable auto scaling.

Perform the following prerequisite steps as necessary:

1. Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
2. From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
• On the Autonomous Databases page select an Autonomous Database from the links under the Display Name column.

1. On the Details page, click Manage Scaling.

2. In the Manage scaling area, add resources for your scale request.
   • Enter a value or click the up arrow to select a value for OCPU count. The default is no change.
   • Enter a value or click the up arrow to select a value for Storage (TB). The default is no change.

3. To enable OCPU auto scaling, select OCPU auto scaling.
Select **OCPU 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 OCPU auto scaling disabled. The additional resources are available until you disable OCPU auto scaling by deselecting **OCPU auto scaling**.

If OCPU auto scaling is disabled while more CPUs are in use than the specified OCPU count, then Autonomous Database scales the number of OCPU cores in use down to the OCPU count number.

See **Use Auto Scaling** for more information.

4. To enable Storage auto scaling, select **Storage auto scaling**.

When you select **Storage auto scaling** the database can expand to use up to three times the reserved base storage.

If you disable **Storage auto scaling** and the used storage is greater than the reserved base storage, as specified by the storage shown in the **Storage** field on the Oracle Cloud Infrastructure Console, Autonomous Database shows a warning on the disable storage auto scaling confirmation dialog. The warning lets you know that the reserved base storage value will be increased to the nearest TB greater than the actual storage usage, and shows the new reserved base storage value.

See **Use Auto Scaling** for more information.

![Note:](image)

**Note:**

Clicking **Shrink** initiates the Shrink storage operation. See **Shrink Storage** for more information.

5. Click **Apply** to change your resources.

When you click **Apply** with a resource change, the Lifecycle State changes to **Scaling in Progress**... After the Lifecycle State changes to **Available** the changes apply immediately.

---

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

Describes how to scale your Autonomous Database on demand by removing CPU cores or storage. Also describes how to disable auto scaling.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the **next to Oracle Cloud.**
  
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
  
- On the Autonomous Databases page select an Autonomous Database from the links under the **Display Name** column.
  
1. On the **Details** page, click **Manage Scaling**.
  
2. In the **Manage scaling** area, select the change in resources for your scale request:
3. When OCPU auto scaling is enabled, deselect **OCPU auto scaling** to disable OCPU auto scaling.

If OCPU auto scaling is disabled while more OCPUs are in use than the specified OCPU count, then Autonomous Database scales the number of OCPU cores in use down to the OCPU count number.

See **Use Auto Scaling** for more information.
4. When Storage auto scaling is enabled, deselect **Storage auto scaling** to disable storage auto scaling.

If you disable **Storage auto scaling** and the used storage is greater than the reserved base storage, as specified by the storage shown in the **Storage** field on the Oracle Cloud Infrastructure Console, Autonomous Database shows a warning on the disable storage auto scaling confirmation dialog. The warning lets you know that the reserved base storage value will be increased to the nearest TB greater than the actual storage usage, and shows the new reserved base storage value.

See Use Auto Scaling for more information.

**Note:**

Clicking **Shrink** initiates the Shrink storage operation. See **Shrink Storage** for more information.

5. Click **Apply** to change your resources.

When you click **Apply** with a resource change, the Lifecycle State changes to **Scaling in Progress**. After the Lifecycle State changes to **Available** the changes apply immediately.

Note the following restrictions for scaling down storage:

- Scaling down storage is not allowed if the Autonomous Database instance contains either of the following:
  - Advanced Queuing tables
  - Tables with **LONG** columns
- If you have columns with the **ROWID** data type, the **ROWIDs** that these column values point to may change during the scale down storage operation.
- Tables that contain the following may be moved offline during a scale down operation. DML operations on these tables may be blocked for the duration of the move and the table indexes for these tables may become unusable until the scale down operation completes:
  - Tables with bitmap join indexes
  - Nested tables
  - Object tables
  - Partitioned tables with domain indexes

---

### Use Auto Scaling

When you create an Autonomous Database instance, by default OCPU auto scaling is enabled and storage auto scaling is disabled. You can manage auto scaling from the Oracle Cloud Infrastructure Console to enable or disable OCPU auto scaling or storage auto scaling.
OCPU Auto Scaling

With OCPU auto scaling enabled the database can use up to three times more CPU and IO resources than specified by the number of OCPUs as shown in the OCPU count on the Oracle Cloud Infrastructure Console.

When OCPU auto scaling is enabled, if your workload requires additional CPU and IO resources, the database automatically uses the resources without any manual intervention required. For example, when the OCPU count is 128, this allows the database to use up to 128 x 3 OCPUs (384 OCPUs) when auto scaling enabled.

To see the average number of OCPUs used during an hour you can use the "Number of OCPUs allocated" graph on the Overview tab in Database Actions (on the Database Monitor card). See Database Monitor Overview for more information.

Enabling OCPU auto scaling does not change the concurrency and parallelism settings for the predefined services. See Manage Concurrency and Priorities on Autonomous Database for more information.

Note:

If your license type is Bring Your Own License (BYOL) with Oracle Database Standard Edition (SE), auto scaling allows the system to automatically use up to three times more CPU and IO resources, capped at a maximum of 8 OCPUs.

See Add CPU or Storage Resources or Enable Auto Scaling for the steps to enable OCPU auto scaling.

Storage Auto Scaling

When you create an Autonomous Database instance, by default Storage auto scaling is disabled. You can manage scaling and enable storage auto scaling from the Oracle Cloud Infrastructure Console.

With Storage auto scaling enabled the Autonomous Database can expand to use up to three times the reserved base storage, as specified by the storage shown in the Storage field on the Oracle Cloud Infrastructure Console. If you need additional storage, the database automatically uses the reserved storage without any manual intervention required.

For example, if your reserved base storage is 128 TB, you have access to 384 TB of storage.

As data flows in, you are billed based on your base storage. For example, if your reserved base storage is 4 TB, after you exceed 4 TB, storage is billed based on the average allocated storage rounded up to the nearest TB per hour (billed by the minute). Thus, for this example with 4 TB of base storage, using storage auto scaling you are charged for 4 TB until you cross 4 TB of allocated storage. As the allocated storage grows over 4 TB, for example up to 4.9 TB, you are charged for 5 TB (based on the average allocated storage billed by the minute).

If you delete 1 TB of data, your allocated storage remains at 4.9 TB and you are billed for 5 TB until you perform a shrink operation. When you perform a shrink operation, Autonomous Database reduces the data and undo tablespace size, in this case back to 3.9 TB. After the
shrink operation completes, billed is again based on your reserved base storage of 4 TB. See Shrink Storage for more information.

Note:
Reducing temp tablespace requires a database restart.

If you disable Storage auto scaling and the used storage is greater than the reserved base storage, as specified by the storage shown in the Storage field on the Oracle Cloud Infrastructure Console, Autonomous Database shows a warning on the disable storage auto scaling confirmation dialog. The warning lets you know that the reserved base storage value will be increased to the nearest TB greater than the actual storage usage, and shows the new reserved base storage value.

To see the Autonomous Database instance storage usage, you can view the "Storage allocated" and "Storage used" graphs on the Overview tab by clicking the Database Monitor card in Database Actions. See Database Monitor Overview for more information.

See Add CPU or Storage Resources or Enable Auto Scaling for the steps to enable storage auto scaling.

Shrink Storage

When the storage used in the database is significantly lower than the allocated storage, the shrink operation reduces the allocated storage.

To understand storage allocation and the shrink operation, note the following:

• Reserved base storage: is the base amount of storage you select for the database when you provision or scale the database, excluding any auto-scaled value. The reserved base storage is shown in the Storage field on the Oracle Cloud Infrastructure Console.

• Allocated storage: is the amount of storage physically reserved for all database tablespaces (excluding sample schema tablespaces). This number also includes the free space in these tablespaces.

• Used storage: is the amount of storage actually used in all tablespaces (excluding the sample schema tablespaces). The used storage excludes the free space in these tablespaces. Used storage is the storage actually used by database objects, tables, indexes, and so on, including internally used temp space.

• Maximum storage: is the maximum storage reserved. When storage auto scaling is disabled, the maximum storage equals the reserved base storage. When storage auto scaling is enabled, the maximum storage is three times the base storage (maximum = reserved base × 3).

Note:
The Shrink operation is not available with Always Free Autonomous Database.
To shrink storage:

1. On the Details page, click Manage Scaling.
2. In the Manage scaling area, select Shrink.

3. Click Confirm in the Shrink Database dialog.

**Note:**

The **Shrink** operation is a long running operation.
The **Shrink** operation requires that all of the following apply:

- **Storage auto scaling** must be enabled.
- The allocated storage must be greater than the reserved base storage.
- The allocated storage, rounded up to the nearest 1TB, can be reduced by 1TB or more.
- The following must be true:
  
  \[
  \text{Allocated storage - Used storage} > 100 \text{ GB}
  \]

When you click **Shrink** and these conditions are not met, Autonomous Database shows the **Action unavailable** dialog.

Note the following restrictions for the **Shrink** operation:

- The shrink operation is not allowed if the Autonomous Database instance contains either of the following:
  - Advanced Queuing tables
  - Tables with **LONG** columns
- If you have columns with the **ROWID** data type, the **ROWIDs** that these column values point to may change during the shrink operation.
- Tables that contain the following may be moved offline during the shrink operation. DML operations on these tables may be blocked for the duration of the move and the table indexes for these tables may become unusable until the shrink operation completes:
  - Tables with bitmap join indexes
  - Nested tables
  - Object tables
  - Partitioned tables with domain indexes

### Schedule Start and Stop Times for an Autonomous Database Instance

Describes the steps to schedule daily start and stop times for an Autonomous Database instance.

When you enable Auto Start/Stop Schedule on an Autonomous Database instance, the instance automatically starts and stops according to the schedule you specify. This allows you to reduce costs by scheduling shutdown periods for times when a system is not in use.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the " next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
• On the Autonomous Databases page select an Autonomous Database from the links under the **Display Name** column.

To set the Auto Start/Stop Schedule, do the following:

1. On the **Details** page, from the **More Actions** drop-down list, select **Auto Start/Stop Schedule**.

2. For each day you want to schedule, select the start time check box and stop time check box and select a start time and a stop time.

   ![Auto Start/Stop Schedule](image)

   The drop-down list allows you to schedule start times and stop times hourly, on the hour, specified in Coordinated Universal Time (UTC).

   Selecting both a start time and a stop time is not required. You can select just a start time or just a stop time.

3. Click **Apply** to apply the schedule.

4. In the confirmation dialog, click **Apply**.

   While the system applies the Auto Start/Stop Schedule, the lifecycle state changes to **Updating**. When the operation completes the lifecycle state shows **Available**.

   The **Auto Start/Stop Schedule** field under General Information on the Autonomous Database Details page lists the days with an Auto Start/Stop Schedule enabled, and includes an **Edit** link.

   To remove start and stop times for a day from an Auto Start/Stop Schedule, edit the schedule and deselect the schedule for that day. To remove all scheduled start and stop times and disable **Auto Start/Stop Schedule**, deselect the start and stop check boxes for all days in the schedule.
Notes for Autonomous Database Auto Start/Stop Schedule:

- A scheduled stop works the same way as a manual stop; when the scheduled stop occurs, any running database sessions are stopped.
- You can perform a manual start for a database, if necessary, after a database has stopped due to a scheduled stop.
  See Start Autonomous Database for more information.
- You can perform a manual stop for a database, if necessary, after a database has started due to a scheduled start.
  See Stop Autonomous Database for more information.
- The Lifecycle State must be Available to view or to edit the Auto Start/Stop Schedule.
- Always Free Autonomous Database does not support Auto Start/Stop Scheduling.
- A database that has been scheduled stopped, remains stopped until the next scheduled start time or until you start the database. Likewise, a database that has been scheduled started, remains running until the next scheduled stop time or until you stop the database.

Change Autonomous Database Operation Mode to Read/Write Read-Only or Restricted

You can select an Autonomous Database operation mode: Read/Write, Read-Only, or Restricted. The default mode is Read/Write.

Topics

- Change Autonomous Database Operation Mode to Read/Write Read-Only or Restricted
- Change Autonomous Database Operation Mode for a Session
- Autonomous Database Operation in Read-Only Mode

Change Autonomous Database Operation Mode to Read/Write Read-Only or Restricted

From the Oracle Cloud Infrastructure Console you can select an Autonomous Database operation mode: Read/Write, Read-Only, or Restricted. The default mode is Read/Write.

If you select Read-Only mode users can only run queries. In addition, for Read-Only or Read/Write mode, you can restrict access to only allow users with the RESTRICTED SESSION privilege to connect to the database. The ADMIN user has this privilege. You can use the restricted access mode to perform administrative tasks such as indexing, data loads, or other planned activities.

Perform the following prerequisite steps as necessary:

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

• On the Autonomous Databases page select an Autonomous Database from the links under the **Display Name** column.

To change the Autonomous Database operation mode, do the following:

1. On the **Details** page, in the **Mode** field under General Information, click **Edit**. This shows the Change Database Mode dialog.

   ![Change Database Mode dialog](image)

2. Select a mode, either **Read/Write** or **Read-Only**.

3. If you want to restrict access to only allow the ADMIN or privileged users, then select **Allow administrator access only**.

4. Click **Confirm**.

   When concurrent operations such as scaling or creating a manual backup are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

   While the mode changes, the Lifecycle State field shows **Updating**. After the mode change completes, the Lifecycle State field shows **Available**.

Notes for changing the database mode:

• The database must be available with the Lifecycle State field showing **Available** to change the database mode.

• When you change the database mode to restricted mode or when you change from restricted mode to unrestricted mode, the database terminates existing database connections as follows:
  
  – When the mode is restricted, that is the Mode field shows **(Admin-only access)**, if you deselect **Allow administrator access only** and click **Confirm**, then users and applications need to reestablish database connections after the mode change completes.

  – When the mode is unrestricted, that is the Mode field does not show **(Admin-only access)**, if you select **Allow administrator access only** and click **Confirm**, then
users and applications need to reestablish database connections after the mode change completes.

**Change Autonomous Database Operation Mode for a Session**

You can set the Autonomous Database operation mode for a session to Read-Only. When you set the session mode to Read-Only users in the session can only run queries.

To enable Read-Only mode for a session:

1. Connect to Autonomous Database.
   
   You can connect as the ADMIN user or as the user whose session you want to set to Read-only mode.

2. Run the following SQL statement:

   `ALTER SESSION SET READ_ONLY = TRUE;`

To disable Read-Only mode for a session:

`ALTER SESSION SET READ_ONLY = FALSE;`

**Note:**

The `READ_ONLY` session parameter applies when the database is in Read/Write mode. If the database is in Read-Only mode, setting the session parameter value to `TRUE` or to `FALSE` does not change the operation mode for the session.

**Autonomous Database Operation in Read-Only Mode**

When the database is in Read-Only mode, each of the following is disabled:

- On the Oracle Cloud Infrastructure Console, the **Administrator Password** action is not allowed. Also, in Database Actions, under Administration, the **Database Users** card is disabled.

- Oracle APEX URLs are not available:
  - In Database Actions, under Development, the APEX link is disabled.
  - On the Oracle Cloud Infrastructure Console Tools tab, the **Open APEX** link is disabled.

- Oracle Machine Learning User Administration is disabled:
  - In Database Actions, under Development, the Oracle Machine Learning link is disabled.
  - On the Oracle Cloud Infrastructure Console Tools tab, the **Open Oracle ML User Administration** link is disabled.

- Setting resource management rules is disabled. In Database Actions, under Administration, the **Set Resource Management Rules** link is disabled.
- Upgrading the database is disabled when the database is in Read-Only mode.

Use Operations Insights on Autonomous Database

Operations Insights is a cloud-native service that enables users to make informed, data-driven, Autonomous Database resource and performance management decisions.

See About Oracle Cloud Infrastructure Operations Insights for details on Operations Insights.

See To enable or disable Operations Insights on an Autonomous Database for information about enabling and disabling Operations Insights.

Use Oracle Data Safe to Assess and Protect Data with Autonomous Database

You can use Oracle Data Safe to identify and protect sensitive and regulated data in Autonomous Databases.

To use Oracle Data Safe you register your database with Oracle Data Safe. Then, you can go to the Oracle Data Safe console from the Details page of your Autonomous Database.

Oracle Data Safe helps you understand the sensitivity of your data, evaluate risks to data, mask sensitive data, implement and monitor security controls, assess user security, monitor user activity, and address data security compliance requirements. It provides the following set of features in a single, easy-to-use management console:

- **Security Assessment** helps you assess the security of your database configuration.
- **User Assessment** helps you assess the security of your database users and identify high risk users.
- **Data Discovery** helps you find sensitive data in your database.
- **Data Masking** provides a way for you to mask sensitive data so that the data is safe for non-production purposes.
- **Activity Auditing** lets you audit user activity on your database so you can monitor database usage and be alerted of unusual database activities.

See Enable Oracle Data Safe for the steps to enable or disable Oracle Data Safe.

For more information about Oracle Data Safe, see Oracle Data Safe Overview.

Concurrent Operations on Autonomous Database

Describes the actions that cause Oracle Cloud Infrastructure Console to ask you to confirm pausing or canceling a concurrent operation.

When you initiate certain operations that take some time to complete, including scaling the system, scaling the OCPUs count or storage up or down, or performing a manual backup, these operations do not prevent you from performing other operations. For example, database connections, database operations, and most Autonomous Database actions proceed normally while you scale the system or when a manual backup is in progress. However, certain database lifecycle management actions such as stopping the database have an impact on concurrent long-running operations.
For some actions a confirmation dialog displays to let you know when a concurrent operation is active. This allows you to proceed or to cancel the operation. For example, while stopping Autonomous Database you see a message such as the following:

**Confirm Stop**

Stopping the database will impact concurrent running operations. [What is the impact?](#)
Are you sure you want to stop the Autonomous Database?

[Stop] [Cancel]

Manage Scaling with Concurrent Operations

Describes the impact on scaling operations when you initiate certain actions during an ongoing scaling request.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>Stop pauses scaling. The scaling restarts when the database starts.</td>
</tr>
<tr>
<td>Restart</td>
<td>Restart pauses scaling. The scaling restarts when the database starts.</td>
</tr>
<tr>
<td>Restore</td>
<td>Restore pauses scaling. The scaling restarts when the restore completes if resources allow.</td>
</tr>
<tr>
<td>Switchover</td>
<td>Switchover pauses scaling. The scaling restarts when the switchover completes. The scaling also occurs on the standby database after the switchover.</td>
</tr>
<tr>
<td>Failover</td>
<td>Failover pauses scaling. The scaling restarts when the failover completes. The scaling also occurs on the standby database if it is available after the failover.</td>
</tr>
<tr>
<td>Terminate</td>
<td>Scaling stops and the database is terminated.</td>
</tr>
<tr>
<td>Change Database Mode</td>
<td>A mode change pauses scaling. The scaling restarts when the database starts after the mode change completes.</td>
</tr>
</tbody>
</table>

Manual Backup with Concurrent Operations

Describes the impact on a manual backup operation when you initiate certain actions during an ongoing manual backup.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>Stop cancels an ongoing manual backup. The manual backup does not restart when the database starts.</td>
</tr>
<tr>
<td>Restart</td>
<td>Restart cancels an ongoing manual backup. The manual backup does not restart when the database starts.</td>
</tr>
<tr>
<td>Action</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Restore</td>
<td>Restore cancels an ongoing manual backup.</td>
</tr>
<tr>
<td>Switchover</td>
<td>Switchover cancels an ongoing manual backup. The manual backup does not restart after the switchover completes.</td>
</tr>
<tr>
<td>Failover</td>
<td>Failover cancels an ongoing manual backup. The manual backup does not restart after the failover completes.</td>
</tr>
<tr>
<td>Terminate</td>
<td>The manual backup is canceled and the database is terminated.</td>
</tr>
</tbody>
</table>
Managing Users on Autonomous Database

This section describes administration tasks for managing users on Autonomous Database.

Topics

• Create Users on Autonomous Database
• Manage User Roles and Privileges on Autonomous Database
• Remove Users on Autonomous Database
• Manage User Profiles with Autonomous Database
• Manage the Administrator Account on Autonomous Database
• Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database
• Use Identity and Access Management (IAM) Authentication with Autonomous Database
• Use Azure Active Directory (Azure AD) with Autonomous Database
• Use Microsoft Active Directory with Autonomous Database
• Use Kerberos Authentication with Autonomous Database

Create Users on Autonomous Database

There are several options to create users on Autonomous Database. You can use Oracle Database Actions Database Users card or use client-side tools that connect to the database to create database users.

Topics

• Create Users on Autonomous Database with Database Actions
• Create Users on Autonomous Database - Connecting with a Client Tool
• Unlock User Accounts on Autonomous Database
• About User Passwords on Autonomous Database
• Manage the Administrator Account on Autonomous Database

Create Users on Autonomous Database with Database Actions

You can quickly create Autonomous Database users with Database Actions.

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

1. Click the top left  next to Oracle Database Actions.
This shows the Database Actions menu, including Development, Data Tools, Administration, and Monitoring.

2. Under Administration click Database Users.

3. On the Database Users page, in the All Users area click Create User.

4. To create a new user, enter a user name, a password, and enter the password again to confirm the password. Also select any options you want to enable for the user: Graph, OML, or Web Access.

5. Set a value for the Quota on tablespace DATA for the user.

6. If you want to grant roles for the new user, click the Granted Roles tab and select the roles for the user. For example, select DWROLE and CONNECT.

7. Click Create User.

Database Actions shows the User Created confirmation message.

See Manage User Roles and Privileges on Autonomous Database for more information on granting roles and adding or updating privileges for a user.
See The Database Users Page for detailed information on Database Actions Database Users.

If you provide Web Access for the new user, then you need to send a URL to the new user. See Provide Database Actions Access to Database Users for more information.

The administrator needs to provide the credentials wallet to the new user for client-side access. See Connecting to Autonomous Database for more information on client-side access credentials.

**Note:**

Autonomous Database requires strong passwords; the password you specify must meet the default password complexity rules. See About User Passwords on Autonomous Database for more information.

See Create Oracle APEX Workspaces in Autonomous Database for information on creating APEX workspaces.

See Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database to add user accounts for Oracle Machine Learning Notebooks.

### Create Users on Autonomous Database - Connecting with a Client Tool

You can create users by connecting to the database as the ADMIN user using any SQL client tool.

For example, connect using Oracle SQL Developer (see Connect Oracle SQL Developer with a Wallet (mTLS)).

1. Connect as the ADMIN user.
2. Run the following SQL statements:

   ```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.
In addition, IDENTIFIED with the BY VALUES clause is not allowed.

This creates `new_user` with connect privileges. This user can now connect to the database and run queries. To grant additional privileges to users, see Manage User Roles and Privileges on Autonomous Database.

The administrator needs to provide the credentials wallet to the user `new_user`. See Connecting to Autonomous Database for more information on client credentials.
Unlock User Accounts on Autonomous Database

If a user account is locked, as the ADMIN user you can unlock the account.

To unlock an account, connect to your database as the ADMIN user and run the following command:

```
ALTER USER username IDENTIFIED BY password ACCOUNT UNLOCK;
```

See SQL Language Reference for information on the ALTER USER command.

About User Passwords on Autonomous Database

Autonomous Database requires strong passwords; the password you specify for a user must meet the minimum default password complexity rules.

Autonomous Database sets minimum standards for passwords, and the default profile sets parameters to limit the number of failed login attempts.

- The password must be between 12 and 30 characters long and must include at least one uppercase letter, one lowercase letter, and one numeric character.
  
  Note, the password limit is shown as 60 characters in some help tooltip popups. Limit passwords to a maximum of 30 characters.

- The password cannot contain the `username`.

- The password cannot be one of the last four passwords used for the same `username`.

- The password cannot contain the double quote (" ) character.

- The password must not be the same password that is set less than 24 hours ago.

To change the password complexity rules and password parameter values you can alter the default profile or create a new profile and assign it to users. See Manage User Profiles with Autonomous Database for more information.

The following are the Autonomous Database default profile password parameter values:
<table>
<thead>
<tr>
<th>Password Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILED_LOGIN_ATTEMPTS</td>
<td>The maximum times a user can try to log in and fail before locking the account. This limit applies for regular database user accounts.</td>
<td>10</td>
</tr>
<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>30</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 Manage User Profiles with Autonomous Database for information on using `CREATE USER` or `ALTER USER` with a profile clause.

See SQL Language Reference for information on the ALTER USER command.

Manage the Administrator Account on Autonomous Database

You can change the administrator user password and when locked, unlock the administrator user account on Autonomous Database.

Set the ADMIN Password in Autonomous Database

Provides the steps to set the ADMIN password.

From the Oracle Cloud Infrastructure Console, change the password for the ADMIN user by following these steps:

1. On the Details page, from the More Actions drop-down list, select Administrator Password.
2. On the Administrator Password page enter the new password and confirm.
3. Click Update.

Note:

You can also use Database Actions to change the password for the ADMIN user. See Manage Users and User Roles on Autonomous Database - Connecting with Database Actions for more information.

The password for the default administrator account, ADMIN, has the same password complexity rules mentioned in the section Create Users on Autonomous Database - Connecting with a Client Tool.
Unlock the ADMIN Account in Autonomous Database

Shows the steps to unlock the ADMIN user account.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the \[ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the Display Name column.

Use the following steps to unlock the ADMIN account by updating the ADMIN password:

1. On the Details page, from the More Actions drop-down list, select Administrator Password.
2. On the Administrator Password page enter the new password and confirm.
3. Click Update.

This operation unlocks the ADMIN account if it was locked.

The password for the default administrator account, ADMIN, has the same password complexity rules mentioned in the section Create Users on Autonomous Database - Connecting with a Client Tool.

Manage User Roles and Privileges on Autonomous Database

There are several ways to manage user privileges and roles on Autonomous Database. You can use Oracle Database Actions Database Users card or client-side tools to connect to the database to manage privileges and roles.

Topics

- Manage Users and User Roles on Autonomous Database - Connecting with Database Actions
- Manage User Privileges on Autonomous Database - Connecting with a Client Tool

Manage Users and User Roles on Autonomous Database - Connecting with Database Actions

You can manage user roles for Autonomous Database users with Oracle Database Actions. The same steps also let you modify account settings for a user.

First, access Database Actions as the ADMIN user. See Access Database Actions as ADMIN for more information.
1. Click the top left icon next to Oracle Database Actions. This shows the Database Actions menu, including Development and Administration.

2. Under Administration click Database Users.

3. On the Database Users page, in the card for the user you want to modify click the icon to open the context menu for the user, then select Edit.

   This shows the Edit User area with the User tab selected.

   **Note:**
   
   If you want to manage the user's account settings, for example if you want to provide Web Access to provide access to Database Actions, or if you want to lock the user's account, you can do this from the User tab.

4. In the Edit User area, click Granted Roles.

   This displays the Granted Roles tab with a list of available roles and selection boxes. For each role, you can check Granted to grant the role, Admin to permit the user to grant the role to other users, and Default to use the default settings for Granted and Admin.

5. Select the roles you want to grant to the user.

   For example, select CONNECT and DWROLE.

   For each role, you can select Granted to grant the role, Admin to permit the user to grant the role to other users, and Default to use the default settings for Granted and Admin. A new user is granted CONNECT and RESOURCE roles when Web Access is selected.
6. Click **Apply Changes**.

See The Database Users Page for more information on Database Actions Database Users.

See [Create Users on Autonomous Database with Database Actions](#) for information on using Database Actions.

**Manage User Privileges on Autonomous Database - Connecting with a Client Tool**

Autonomous Databases come with a predefined database role named **DWROLE**. This role provides the common privileges for a database developer or 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 Oracle SQL Developer with a Wallet (mTLS)](#)).

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

   ```sql
   GRANT DWROLE TO adb_user;
   ```

3. Grant individual privileges to users with the **GRANT** command instead of or in addition to granting **DWROLE** privileges. See [Oracle Database SQL Language Reference](#).
4. If a user needs to load data, do one of the following to add the privileges required to load data:
   - Add quota to a new user with `CREATE USER` or alter the quota for an existing user with `ALTER USER`. For example:
     ```
     CREATE USER sales
     QUOTA 5M on DATA;
     ALTER USER sales
     QUOTA 1G on DATA;
     ```
   - Grant `UNLIMITED TABLESPACE` privileges to a user. For example, the following command grants unlimited tablespace privileges to the user `adb_user`:
     ```
     GRANT UNLIMITED TABLESPACE TO adb_user;
     ```

   **Note:**
   Granting `UNLIMITED TABLESPACE` privilege allows a user to use all the allocated storage space. You cannot selectively revoke tablespace access from a user with the `UNLIMITED TABLESPACE` privilege. You can grant selective or restricted access only after revoking the privilege.

The privileges in `DWROLE` are the following:

- `CREATE ANALYTIC VIEW`
- `CREATE ATTRIBUTE DIMENSION`
- `ALTER SESSION`
- `CREATE HIERARCHY`
- `CREATE JOB`
- `CREATE MATERIALIZED VIEW`
- `CREATE MINING MODEL`
- `CREATE PROCEDURE`
- `CREATE SEQUENCE`
- `CREATE SESSION`
- `CREATE SYNONYM`
- `CREATE TABLE`
- `CREATE TRIGGER`
- `CREATE TYPE`
- `CREATE VIEW`
- `READ,WRITE ON directory DATA_PUMP_DIR`
- `EXECUTE privilege on the PL/SQL package DBMS_CLOUD`
- `EXECUTE privilege on OCI PL/SQL SDK`

## Remove Users on Autonomous Database

To remove users from your database, connect to the database as the `ADMIN` user using any SQL client tool.

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

1. Connect as the `ADMIN` user.
2. Run the following SQL statement:

```
DROP USER user_name CASCADE;
```

This removes user_name and the objects owned by that user.

**Note:**
This removes all user_name objects and the data owned by user_name is deleted.

You can also remove a database user with Oracle Database Actions. To remove a user, from the Database Users page, click the icon to open the context menu for a user, and then select **Delete**. See Manage Users and User Roles on Autonomous Database - Connecting with Database Actions for more information on using Oracle Database Actions Database Users.

## Manage User Profiles with Autonomous Database

You can create and alter user profiles in Autonomous Database. After you create or alter a profile, you can specify the profile clause with CREATE USER or ALTER USER. You can also import existing user profiles from another environment with Oracle Data Pump Import.

**Note:**
Autonomous Database has restrictions on the profile clause. See SQL Commands for information on CREATE PROFILE and ALTER PROFILE restrictions.

To add, modify, or remove a password parameter in a profile, including the DEFAULT profile you must have the ALTER PROFILE system privilege.

1. To add or alter a profile, as the ADMIN user run either CREATE PROFILE or ALTER PROFILE. For example:

```
CREATE PROFILE new_profile
  LIMIT PASSWORD_REUSE_MAX 10
  PASSWORD_LOCK_TIME 5;
```

**Note:**
If you are not the ADMIN user, then you must have CREATE PROFILE privilege to run CREATE PROFILE. If you run ALTER PROFILE, then you must have ALTER PROFILE privilege.
2. Use the new or altered profile with a CREATE USER or ALTER USER command. For example:

```
CREATE USER new_user IDENTIFIED BY password PROFILE new_profile;
GRANT CREATE SESSION TO new_user;
```

This creates new_user with profile new_profile and with connect privileges. The new_user can now connect to the database and run queries. To grant additional privileges to users, see Manage User Privileges on Autonomous Database - Connecting with a Client Tool.

See CREATE PROFILE for information on using CREATE PROFILE or ALTER PROFILE.

You can import existing profiles created in other environments using Oracle Data Pump Import (impdp). Any existing profile association with database users is preserved after importing into Autonomous Database. When a newly created user, created from an Oracle Data Pump import, attempts to login for the first time, if the user's password violates the password complexity requirements then the account is expired and the user is required to change their password before the user can successfully login to Autonomous Database. These password restrictions are the same as the restrictions for any user on Autonomous Database.

**Note:**

Profile assignments for users with profile ORA_PROTECTED_PROFILE cannot be modified.

When you create or alter a profile, you can specify a Password Verification Function (PVF) to manage password complexity. See Manage Password Complexity on Autonomous Database for more information.

### Manage Password Complexity on Autonomous Database

You can create a Password Verify Function (PVF) and associate the PVF with a profile to manage the complexity of user passwords.

**Note:**

The minimum password length for a user specified PVF is 8 characters. The minimum password length for the DEFAULT profile is 12 characters (the DEFAULT profile uses the CLOUD_VERIFY_FUNCTION PVF).
Oracle recommends using a minimum password length of 12 characters. If you define a profile's PVF, and set the minimum password length to less than 12 characters, then tools such as Oracle Database Security Assessment Tool (DBSAT) and Qualys report this as a database security risk.

For example, to specify a PVF for a profile, use the following command:

```
CREATE PROFILE example_profile LIMIT PASSWORD_VERIFY_FUNCTION ADMIN.EXAMPLE_PVF
```
If the profile is created or altered by any user other than the ADMIN user, then you must grant the EXECUTE privilege on the PVF. If you create a PVF and the password check fails, the database reports the ORA-28219 error.

You can specify an Oracle supplied PVF, from one of the following:

- **CLOUD_VERIFY_FUNCTION** (this is the default password verification function for Autonomous Database):
  This function checks for the following requirements when users create or modify passwords:
  - The password must be between 12 and 30 characters long and must include at least one uppercase letter, one lowercase letter, and one numeric character.
  - The password cannot contain the username.
  - The password cannot be one of the last four passwords used for the same username.
  - The password cannot contain the double quote (") character.
  - The password must not be the same password that is set less than 24 hours ago.

- **ORA12C_STIG_VERIFY_FUNCTION**
  This function checks for the following requirements when users create or modify passwords:
  - The password has at least 15 characters.
  - The password has at least 1 lower case character and at least 1 upper case character.
  - The password has at least 1 digit.
  - The password has at least 1 special character.
  - The password differs from the previous password by at least 8 characters.
  See ora12c_stig_verify_function Password Requirements for more information.

Note the following restrictions for a Password Verification Function (PVF) that you create and assign to a profile:

- If you specify a user profile, the minimum password length depends on how you define the associated PVF, as follows:
  - If a PVF is defined, then the minimum password length enforced is 8 characters.
  - If the PVF is defined as NULL, then the minimum password length enforced is 8 characters.
  - If the profile does not have a PVF defined, then the DEFAULT profile’s PVF (CLOUD_VERIFY_FUNCTION) is assigned and the minimum password length enforced is 12 characters.

- If you specify a Password Verify Function (PVF) that is more strict than the default CLOUD_VERIFY_FUNCTION, then the new verify function is used.

- A PVF that you create must be created as a DEFINER RIGHTS PL/SQL function. If an INVOKER rights PVF is provided as input to CREATE or ALTER PROFILE, then ORA-28220 error is thrown.
• Any PVF that you create must be created in the ADMIN user schema. If a non-ADMIN user owned PVF is provided as input to CREATE or ALTER PROFILE, then ORA-28220 error is thrown.

• A PVF cannot be altered or dropped by a non-ADMIN user. That is, any user with the CREATE or DROP ANY PROCEDURE privilege is not allowed to alter or drop a PVF.

• If the PVF associated with a profile is dropped, then any attempt to change the password for a user who uses the PVF in their profile throws the error ORA-7443. Users can still login when the PVF associated with their profile is dropped. However, if a user's password is expired and the PVF is dropped, then the user cannot login.

To recover from the ORA-7443 error, the ADMIN user must recreate the dropped PVF and assign it to the profile, or assign an existing PVF to the profile. This allows a user change their password and login.

• The CREATE ANY PROCEDURE system privilege and DROP ANY PROCEDURE system privilege are audited for PVF security. See the PROCEDURES list in Listings of System and Object Privileges for more information.

See Managing the Complexity of Passwords for more information.

Gradual Database Password Rollover for Applications

An application can change its database passwords without an administrator having to schedule downtime.

To accomplish this, you can associate a profile having a non-zero limit for the PASSWORD_ROLLOVER_TIME password profile parameter, with an application schema. This allows the database password of the application user to be altered while allowing the older password to remain valid for the time specified by the PASSWORD_ROLLOVER_TIME limit. During the rollover period of time, the application instance can use either the old password or the new password to connect to the database server. When the rollover time expires, only the new password is allowed.

See Managing Gradual Database Password Rollover for Applications for more information.

Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database

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

Topics

• Create User

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

An administrator creates new user accounts and user credentials for Oracle Machine Learning in the User Management interface.

**Note:**
You must have the administrator role to access the Oracle Machine Learning User Management interface.

To create a user account:

1. On the Autonomous Databases page, under the Display Name, select an Autonomous Database.
2. On the Autonomous Database Details page, click Database Actions.
3. On the Database Actions launchpad, under Administration, click Database Users.
4. Click + Create User.
5. In the User Name field, enter a username for the account. Using the username, the user will log in to an Oracle Machine Learning instance.
6. (Optional) Select the option Password Expired (user must change) to required the user to change their password when they login for the first time.
7. In the Password field, enter a password for the user.
8. In the Confirm Password field, enter a password to confirm the value that you entered in the Password field.
9. Select OML to enable Oracle Machine Learning for the user.
10. Click Create User.

This creates a new database user and grants the required privileges to use Oracle Machine Learning.

**Note:**
With a new database user, an administrator needs to issue grant commands on the database to grant table access to the new user for the tables associated with the user's Oracle Machine Learning notebooks.
Add Existing Database User Account to Oracle Machine Learning Components

As the ADMIN user you can add an existing database user account for Oracle Machine Learning components.

**Note:**

You must have the ADMIN role to access the Oracle Machine Learning User Management interface.

To add an existing database user account:

1. On the Autonomous Databases page, under the Display Name column, select an Autonomous Database.
2. On the Autonomous Database Details page, click Database Actions.
4. Expand the navigator by clicking the next to Oracle Machine Learning.
6. Click Show All Users to display the existing database users.

- **Note:** Initially, the Role field shows the role None for existing database users. After adding a user the role Developer is assigned to the user.

7. Select a user. To select a user select a name in the User Name column. For example, select ANALYST1.

Selecting the user shows the Oracle Machine Learning Edit User page.
8. Enter a name in the First Name field. (Optional)
9. Enter the last name of the user in the Last Name field. (Optional)
10. In the Email Address field, enter the email ID of the user.

Making any change on this page adds the existing database user with the required privileges as an Oracle Machine Learning component user.

11. Click Save.

This grants the required privileges to use the Oracle Machine Learning application. In Oracle Machine Learning this user can then access any tables the user has privileges to access in the database.

Use Identity and Access Management (IAM) Authentication with Autonomous Database

You can configure Autonomous Database to use Oracle Cloud Infrastructure Identity and Access Management (IAM) authentication and authorization to allow IAM users to access an Autonomous Database with IAM credentials.

Topics
- About Identity and Access Management (IAM) Authentication with Autonomous Database
- Prerequisites for Identity and Access Management (IAM) Authentication on Autonomous Database
- Enable Identity and Access Management (IAM) Authentication on Autonomous Database
- Create Identity and Access Management (IAM) Groups and Policies for IAM Users
- Add IAM Users on Autonomous Database
- Add IAM Roles on Autonomous Database
- Create IAM Database Password for IAM Users
- Connect to Autonomous Database with Identity and Access Management (IAM) Authentication
- Configure Proxy Authentication
- Disable Identity and Access Management (IAM) Authentication on Autonomous Database
- Notes for Using Autonomous Database Tools with Identity and Access Management (IAM) Authentication

About Identity and Access Management (IAM) Authentication with Autonomous Database

You can enable an Autonomous Database instance to use Oracle Cloud Infrastructure (IAM) authentication and authorization for users.
Autonomous Database integration with Oracle Cloud Infrastructure IAM is supported in commercial tenancies with identity domains as well as the legacy Oracle Cloud Infrastructure IAM, which does not include identity domains. Oracle Cloud Infrastructure IAM with identity domains was introduced with new OCI tenancies created after November 8, 2021. Only default domain OCI IAM users are supported with the new identity domains.

Oracle Cloud Infrastructure IAM integration with Autonomous Database supports the following:

- **IAM Database Password Authentication**
- **Identity and Access Management (IAM) SSO Token Based Authentication**

See Authenticating and Authorizing IAM Users for Oracle Autonomous Databases for complete details about the architecture for using IAM users on Autonomous Database.

**IAM Database Password Authentication**

You can enable an Autonomous Database instance to allow user access with an Oracle Cloud Infrastructure IAM database password (using a password verifier).

**Note:**

Any supported 12c and above database client can be used for IAM database password access to Autonomous Database.

An Oracle Cloud Infrastructure IAM database password allows an IAM user to log in to an Autonomous Database instance as Oracle Database users typically log in with a user name and password. The user enters their IAM user name and IAM database password. An IAM database password is a different password than the Oracle Cloud Infrastructure Console password. Using an IAM user with the password verifier you can login to Autonomous Database with any supported database client.

**Identity and Access Management (IAM) SSO Token Based Authentication**

You can enable an Autonomous Database instance to use Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM) SSO tokens.

There are several ways a database client can obtain an IAM database token:

- A client application or tool can request the database token from IAM for the user and can pass the database token through the client API. Using the API to send the token overrides other settings in the database client. This type of IAM database token usage is supported for the following clients:
  - JDBC-thin on all platforms
  - Oracle Instant Client OCI-C on Linux
  - Oracle Data Provider for .NET (ODP.NET) Core
• If the application or tool does not support requesting an IAM database token through the client API, the IAM user can first use Oracle Cloud Infrastructure command line interface (CLI) to retrieve the IAM database token and save it in a file location. For example, to use SQL*Plus and other applications and tools using this connection method, you first obtain the database token using the Oracle Cloud Infrastructure (OCI) Command Line Interface (CLI). If the database client is configured for IAM database tokens, when a user logs in with the slash login form, the database driver uses the IAM database token that has been saved in a default or specified file location.

• A client application or tool can use an Oracle Cloud Infrastructure IAM instance principal or resource principal to get an IAM database token, and use the IAM database token to authenticate itself to an Autonomous Database instance.

• IAM users and OCI applications can request a database token from IAM with several methods, including using an API-key. See Configuring a Client Connection for SQL*Plus That Uses an IAM Token for an example. See About Authenticating and Authorizing IAM Users for an Oracle Autonomous Database for a description of other methods such as using a delegation token within an OCI cloud shell.

If a user enters a username/password to login, then the database driver uses the password verifier method to access the database regardless of the client’s database token setting.

Prerequisites for Identity and Access Management (IAM) Authentication on Autonomous Database

Describes the prerequisites for enabling IAM user access on Autonomous Database.

If the database is enabled for another external authentication scheme, verify that you want to use IAM on the Autonomous Database instance. There can only be one external authentication scheme enabled at any given time.

If you want to use IAM and another external authentication scheme is enabled, you can either first disable the other external authentication scheme or use the `force` parameter with the value set to true when you enable IAM authentication. See Enable Identity and Access Management (IAM) Authentication on Autonomous Database for information on using the `force` parameter.

Enable Identity and Access Management (IAM) Authentication on Autonomous Database

Describes the steps to enable IAM user access on Autonomous Database.

---

**Note:**

Autonomous Database integration with Oracle Cloud Infrastructure IAM is supported in commercial tenancies with identity domains as well as the legacy Oracle Cloud Infrastructure IAM, which does not include identity domains. Oracle Cloud Infrastructure IAM with identity domains was introduced with new OCI tenancies created after November 8, 2021. Only default domain OCI IAM users are supported with the new identity domains.
To enable Autonomous Database to allow IAM users to connect to the database:

1. Perform the prerequisites for IAM authorization and authentication on Autonomous Database. See Prerequisites for Identity and Access Management (IAM) Authentication on Autonomous Database for more information.

2. Use the procedure `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION` to enable Oracle Cloud Infrastructure IAM authentication.

   When you perform these steps, connect to the Autonomous Database instance as the ADMIN user or as a user with ADMIN privileges.

   For example:

   ```
   BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
     type => 'OCI_IAM');
   END;
   /
   ```

   By default the `force` parameter is false. When another external authentication method is enabled and `force` is false, `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION` reports the following error:

   `ORA-20004: Another external authentication is already enabled.`

   If you want to disable the external authentication that is currently enabled and use IAM authentication instead, include the `force` parameter.

   For example:

   ```
   BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
     type =>'OCI_IAM',
     force => TRUE);
   END;
   /
   ```

   This sets the `IDENTITY_PROVIDER_TYPE` system parameter.

   For example, you can use the following to verify `IDENTITY_PROVIDER_TYPE`:

   ```
   SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME='identity_provider_type';
   ```

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>identity_provider_type</td>
<td>OCI_IAM</td>
</tr>
</tbody>
</table>
Create Identity and Access Management (IAM) Groups and Policies for IAM Users

Describes the steps to write policy statements for an IAM group to enable IAM user access to Oracle Cloud Infrastructure resources, specifically Autonomous Database instances.

A policy is a group of statements that specifies who can access particular resources, and how. Access can be granted for the entire tenancy, databases in a compartment, or individual databases. This means you write a policy statement that gives a specific group a specific type of access to a specific type of resource within a specific compartment.

Note:

Defining a policy is required to use IAM tokens to access Autonomous Database. A policy is not required when using IAM database passwords to access Autonomous Database.

To enable Autonomous Database to allow IAM users to connect to the database using IAM tokens:

1. Perform Oracle Cloud Infrastructure Identity and Access Management prerequisites by creating a group and adding users to the group.
   For example, create the group sales_dbusers.
   See Managing Groups for more information.

2. Write policy statements to enable access to Oracle Cloud Infrastructure resources.
   a. In the Oracle Cloud Infrastructure console click Identity and Security and click Policies.
   b. To write a policy, click Create Policy, and enter a Name and a Description.
   c. Use the Policy Builder to create a policy.
      For example to create a policy to allow users in IAM group DBUsers to access any Autonomous Database in their tenancy:

      Allow group DBUsers to use autonomous-database-family in tenancy

      For example to create a policy that limits members of DBUsers group to access Autonomous Databases in compartment testing_compartment only:

      allow group DBUsers to use autonomous-database-family in compartment testing_compartment
For example to create a policy that limits group access to a single database in a compartment:

allow group DBUsers to use autonomous-database-family in compartment testing_compartment where target.database.id = 'ocid1.autonomousdatabase.oc1.iad.aaaabbbbcccc'

d. Click Create.

See Managing Policies for more information on policies.

Notes for creating policies for use with IAM users on Autonomous Database:

- Policies can allow IAM users to access Autonomous Database instances across the entire tenancy, in a compartment, or can limit access to a single Autonomous Database instance.

- You can use either instance principal or resource principal to retrieve database tokens to establish a connection from your application to an Autonomous Database instance. If you are using an instance principal or resource principal, you must map a dynamic group. Thus, you cannot exclusively map instance and resource principals; you only can map them through a shared mapping and putting the instance or resource instance in an IAM dynamic group.

You can create Dynamic Groups and reference dynamic groups in the policies you create to access Oracle Cloud Infrastructure. See Accessing Cloud Resources by Configuring Policies and Roles and Managing Dynamic Groups for details.

Add IAM Users on Autonomous Database

To add IAM users to allow access to Autonomous Database, map database global users to IAM groups or users with CREATE USER or ALTER USER statements (with IDENTIFIED GLOBALLY AS clause).

The authorization of IAM users to an Autonomous Database instance works by mapping IAM global users (schemas) to IAM users (exclusive mapping) or IAM groups (shared schema mapping).

To authorize IAM users on an Autonomous Database instance:

1. Log in as the ADMIN user to the database that is enabled to use IAM (the ADMIN user has the required CREATE USER and ALTER USER system privileges that you need for these steps).

2. Create a mapping between the Autonomous Database user (schema) with CREATE USER or ALTER USER statements and include the IDENTIFIED GLOBALLY AS clause, specifying the IAM group name.

Use the following syntax to map a global user to an IAM group:

CREATE USER global_user IDENTIFIED GLOBALLY AS 'IAM_GROUP_NAME=IAM_GROUP_NAME';
For example, to map an IAM group named `db_sales_group` to a shared database global user named `sales_group`:

```
CREATE USER sales_group IDENTIFIED GLOBALLY AS 'IAM_GROUP_NAME=db_sales_group';
```

This creates a shared global user mapping. The mapping, with global user `sales_group`, is effective for all users in the IAM group. Thus, anyone in the `db_sales_group` can log in to the database using their IAM credentials (through the shared mapping of the `sales_group` global user).

3. If you want to create additional global user mappings for other IAM groups or users, follow these steps for each IAM group or user.

Note:

Database users that are not IDENTIFIED GLOBALLY can continue to login as before, even when the Autonomous Database is enabled for IAM authentication.

To exclusively map a local IAM user to an Oracle Database Global User:

1. Log in as the ADMIN user to the database that is enabled to use IAM (the ADMIN user has the required CREATE USER and ALTER USER system privileges that you need for these steps).

2. Create a mapping between the Autonomous Database user (schema) with CREATE USER or ALTER USER statements and include the IDENTIFIED GLOBALLY AS clause, specifying the IAM local IAM user name.

   For example, to create a new database global user named `peter_fitch` and map this user to an existing local IAM user named `peterfitch`:

   ```
   CREATE USER peter_fitch IDENTIFIED GLOBALLY AS 'IAM_PRINCIPAL_NAME=peterfitch'
   ```

Add IAM Roles on Autonomous Database

Optionally, create global roles to provide additional database roles and privileges to IAM users when multiple IAM users are mapped to the same shared global user.

Creating global roles is optional for an IAM user with an exclusive IAM mapping to an Autonomous Database user (schema). When the IAM mapping is to a shared schema, creating a global role is also optional. For example, all privileges and roles can be granted to the shared schema and all IAM users who map to the shared schema would be granted the privileges and roles assigned to the shared schema.

Use a global role to optionally differentiate users who use the same shared schema. For example, a set of users can all have the same shared schema and the shared schema could have the CREATE SESSION privilege. Then global roles can be used to provide differentiated privileges and roles assigned to different groups of users who all use the same shared schema.
Granting additional roles to IAM users in Autonomous Database works by mapping Autonomous Database global roles to IAM groups.

To map Autonomous Database global roles to IAM groups:

1. Log in as the ADMIN user to the database that is enabled to use IAM (the ADMIN user has the required CREATE USER and ALTER USER system privileges that you need for these steps).

2. Set database authorization for Autonomous Database roles with CREATE ROLE or ALTER ROLE statements and include the IDENTIFIED GLOBALLY AS clause, specifying the IAM group name.

   Use the following syntax to map a global role to an IAM group:

   ```sql
   CREATE ROLE global_role IDENTIFIED GLOBALLY AS 'IAM_GROUP_NAME=IAM_GROUP_of_WHICH_the_IAM_USER_IS_a_MEMBER';
   ```

   For example, to map an IAM group named ExporterGroup to a shared database global role named export_role:

   ```sql
   CREATE ROLE export_role IDENTIFIED GLOBALLY AS 'IAM_GROUP_NAME=ExporterGroup';
   ```

3. Use GRANT statements to grant the required privileges or other roles to the global role.

   ```sql
   GRANT CREATE SESSION TO export_role;
   GRANT DWROLE TO export_role;
   ```

4. If you want an existing database role to be associated with an IAM group, then use ALTER ROLE statement to alter the existing database role to map the role to an IAM group. Use the following syntax to alter an existing database role to map it to an IAM group:

   ```sql
   ALTER ROLE existing_database_role IDENTIFIED GLOBALLY AS 'IAM_GROUP_NAME=IAM_Group_Name';
   ```

   If you want to add additional global role mappings for other IAM groups, follow these steps for each IAM group.

Create IAM Database Password for IAM Users

To add an IAM user and allow the IAM user to login to Autonomous Database by supplying a username and password, you must create an IAM database password. You do not need to create an IAM database password for IAM token based access.

See Working with IAM Database Passwords for more information.

Connect to Autonomous Database with Identity and Access Management (IAM) Authentication

After the ADMIN user enables Oracle Cloud Infrastructure IAM on Autonomous Database, users log in to the Autonomous Database instance using their Oracle Cloud Infrastructure
IAM credentials or access the database through an Oracle Cloud Infrastructure IAM database token.

After you enable Oracle Cloud Infrastructure IAM user access, you can also log in to the Autonomous Database using your local database account username and password (non-global database user account).

You can use a database client to access an Autonomous Database instance as an Oracle Cloud Infrastructure IAM user. To use a client with Oracle Cloud Infrastructure IAM username and password credentials and a password verifier, the database client must be 12c or newer or older clients must be patched with the 12c password verifier.

Alternatively, you can use an Oracle Cloud Infrastructure IAM database token to access an Autonomous Database instance with supported clients:

- JDBC-Thin with support for IAM Token Authentication is supported with the following:
  - JDBC version 19.13.0.0.1 (or later): See JDBC and UCP Downloads for JDBC drivers.
  - JDBC version 21.4.0.0.1 (or later): See JDBC and UCP Downloads for JDBC drivers.

See Support for IAM Token-Based Authentication for more information:

- SQL*Plus and Oracle Instant Client: Supported with SQL*Plus and Instant Client on Linux versions 19.13 or later, and Instant Client on Linux versions 21.4 or later. See Identity and Access Management (IAM) Token-Based Authentication for more information.

- .NET clients (latest version of Linux or Windows). .NET software components are available as a free download from the following sites:
  - Oracle Data Access Components - .NET Downloads
  - NuGet Gallery
  - Visual Studio Code Marketplace

The following examples show password verifier with SQL*Plus to access the database with an Oracle Cloud Infrastructure IAM username and password and the steps required to use SQL*Plus with an Oracle Cloud Infrastructure IAM database token.

Note:

If your Autonomous Database instance is in Restricted Mode, only the users with the RESTRICTED SESSION privilege such as ADMIN can connect to the database.

About Connecting to an Oracle Autonomous Database Instance Using IAM

IAM users can connect to the Oracle Autonomous Database instance by using either an IAM database password verifier or an IAM token.

Using the IAM database password verifier is similar to the Oracle Database password authentication process. However, instead of the password verifier (encrypted hash of
the password) being stored in the Oracle Autonomous Database, the verifier is instead stored as part of the Oracle Cloud Infrastructure (OCI) IAM user profile.

The second connection method, the use of an IAM token for the database, is more modern. The use of token-based access is a better fit for Cloud resources such as Oracle Autonomous Database. The token is based on the strength that the IAM endpoint can enforce. This can be multi-factor authentication, which is stronger than the use of passwords alone. Another benefit of using tokens is that the password verifier (which is considered sensitive) is never stored or available in memory.

Client Connections That Use an IAM Database Password Verifier

After you have configured the authorization needed for the IAM user, this user can log in using existing client application, such as SQL*Plus or SQLcl without additional configuration.

The IAM user enters the IAM user name and IAM database password (not the Oracle Cloud Infrastructure console password) using any currently supported database client. The only constraint is that the database client version be either Oracle Database release 12.1.0.2 or later (or patched) to allow Oracle Database 12c passwords. The database client must be able to use the 12c password verifier. Using the 11G verifier encryption is not supported with IAM. No special client or tool configuration is needed for the IAM user to connect to the Oracle Autonomous Database instance.

Client Connections That Use a Token

For IAM token access to the Oracle Autonomous Database, the client application or tool requests a database token from IAM for the IAM user.

The client application will pass the database token directly to the database client through the database client API.

If the application or tool has not been updated to request an IAM token, then the IAM user can use Oracle Cloud Infrastructure (OCI) command line interface (CLI) to request and store the database token. You can request a database access token (db-token) using the following credentials:

- Security tokens (with IAM authentication), delegation tokens (in the OCI cloud shell) and API-keys, which are credentials that represent the IAM user to enable the authentication
- Instance principal tokens, which enable instances to be authorized actors (or principals) to perform actions on service resources after authenticating
- Resource principal token, which is a credential that enables the application to authenticate itself to other Oracle Cloud Infrastructure services

When the IAM users logs into the client with a slash / login and the OCI_IAM parameter is configured (sqlnet.ora, tnsnames.ora, or as part of a connect string), then the database client retrieves the database token from a file. If the IAM user submits a user name and password, the connection will use the IAM database verifier access described for client connections that use IAM database password verifiers. The instructions in this guide show how to use the OCI CLI as a helper for the database token. If the application or tool has been updated to work with IAM, then follow the instructions for the application or tool. Some common use cases include the following: SQLPlus on-premises, SQLcl on-premises, SQL*Plus in Cloud Shell, or applications that use SEP wallets.
Configuring a Client Connection for SQL*Plus That Uses an IAM Database Password

You can configure SQL*Plus to use an IAM database password.

- As the IAM user, log in to the Oracle Autonomous Database by using the following syntax:

  ```
  CONNECT user_name@db_connect_string
  Enter password: password
  ```

  In this specification, `user_name` is the IAM user name. There is a limit of 128 bytes for the combined `domain_name/user_name`.

  The following example shows how IAM user `peter_fitch` can log in to an Oracle Autonomous Database instance.

  ```
  sqlplus /nolog
  connect peter_fitch@db_connect_string
  Enter password: password
  ```

  Some special characters will require double quotation marks around `user_name` and `password`. For example:

  ```
  "peter_fitch@example.com"@db_connect_string
  "IAM database password"
  ```

Configuring a Client Connection for SQL*Plus That Uses an IAM Token

You can configure a client connection for SQL*Plus that uses an IAM token.

1. Ensure you have an IAM user account.
2. Check with an IAM administrator and Autonomous Database administrator to ensure you have a policy allowing you to access the database in the compartment or your tenancy and that you are mapped to a global schema in the database.
3. If your application or tool does not support direct IAM integration, then download, install, and configure the OCI CLI. (See [OCI Command Line Interface Quickstart](#).) Set up an API key as part of the OCI CLI configuration and select default values.
   a. Set up the API key access for the IAM user.
   b. Retrieve the `db-token`. For example:

      ```
      Retrieving a `db-token` with an API-key using the Oracle Cloud Infrastructure (OCI) command-line interface:
      ```

      ```
      oci iam db-token get
      ```
• Retrieving a db-token with a security (or session) token:

oci iam db-token get --auth security_token

If the security token has expired, a window will appear so the user can log in to OCI again. This generates the security token for the user. OCI CLI will use this refreshed token to get the db-token.

• Retrieving a db-token with a delegation token: When you log in to the cloud shell, the delegation token is automatically generated and placed in the /etc directory. To get this token, execute the following command in the cloud shell:

oci iam db-token get

• Retrieving an instance token by using the OCI command-line interface:

oci iam db-token get --auth instance_principal

See Required Keys and OCIDs for more information.

4. Ensure that you are using the latest release updates for the Oracle Database client releases 19c and 21c.

This configuration only works with the Oracle Database client release 19c or 21c.

5. Follow the existing process to download the wallet from the Oracle Autonomous Database and then follow the directions for configuring it for use with SQL*Plus.

a. Confirm that DN matching is enabled by looking for SSL_SERVER_DN_MATCH=ON in sqlnet.ora.

b. Configure the database client to use the IAM token by adding TOKEN_AUTH=OCI_TOKEN to the sqlnet.ora file. Because you will be using the default locations for the database token file, you do not need to include the token location.

The TOKEN_AUTH and TOKEN_LOCATION values in the tnsnames.ora connect strings take precedence over the sqlnet.ora settings for that connection. For example, for the connect string, assuming that the token is in the default location (~/.oci/db-token for Linux):

(description=
  (retry_count=20)(retry_delay=3)
  (address=(protocol=tcp$)(port=1522)
  (host=example.us-phoenix-1.oraclecloud.com)}

  (connect_data=(service_name=aaabbbcc_exampledb_high.example.oraclecloud.com))

  (security=(ssl_server_cert_dn="CN=example.uscom-east-1.oraclecloud.com,
    OU=Oracle BMCS US, O=Example Corporation,
    L=Redwood City, ST=California, C=US")
  (TOKEN_AUTH=OCI_TOKEN))}

After the connect string is updated with the TOKEN_AUTH parameter, the IAM user can log in to the Oracle Autonomous Database instance by running the following command to start
SQL*Plus. You can include the connect descriptor itself or use the name of the descriptor from the tnsnames.ora file.

connect /@exampledb_high

Or:

connect /@{(description=(retry_count=20)(retry_delay=3)(address=(protocol=tcps)(port=1522)(host=example.us-phoenix-1.oraclecloud.com))(connect_data=(service_name=aaabbbccc_exampledb_high.example.oraclecloud.com))(security=(ssl_server_cert_dn="CN=example.uscom-east-1.oraclecloud.com, OU=Oracle BMCS US, O=Example Corporation, L=Redwood City, ST=California, C=US") (TOKEN_AUTH=OCI_TOKEN)))}

The database client is already configured to get a db-token because TOKEN_AUTH has already been set, either through the sqlnet.ora file or in a connect string. The database client gets the db-token and signs it using the private key and then sends the token to the Oracle Autonomous Database. If an IAM user name and IAM database password are specified instead of slash /, then the database client will connect using the password instead of using the db-token.

Use Instance Principal to Access Autonomous Database with Identity and Access Management (IAM) Authentication

After the ADMIN user enables Oracle Cloud Infrastructure IAM on Autonomous Database, an application can access the database through an Oracle Cloud Infrastructure IAM database token using an instance principal. See Accessing the Oracle Cloud Infrastructure API Using Instance Principals for more information.

Configure Proxy Authentication

Proxy authentication allows an IAM user to proxy to a database schema for tasks such as application maintenance.

About Configuring Proxy Authentication

IAM users can connect to Oracle Autonomous Database by using proxy authentication.

Proxy authentication is typically used to authenticate the real user and then authorize them to use a database schema with the schema privileges and roles in order to manage an application. Alternatives such as sharing the application schema password are considered insecure and unable to audit which actual user performed an action.
A use case can be in an environment in which a named IAM user who is an application
database administrator can authenticate by using their credentials and then proxy to a
database schema user (for example, hrapp). This authentication enables the IAM
administrator to use the hrapp privileges and roles as user hrapp in order to perform
application maintenance, yet still use their IAM credentials for authentication. An application
database administrator can sign in to the database and then proxy to an application schema
to manage this schema.

You can configure proxy authentication for both the password authentication and token
authentication methods.

**Configure Proxy Authentication for the IAM User**

To configure proxy authentication for an IAM user, the IAM user must already have a mapping
to a global schema (exclusive or shared mapping). A separate database schema for the IAM
user to proxy to must also be available.

After you ensure that you have this type of user, alter the database user to allow the IAM user
to proxy to it.

1. Log in to the Autonomous Database instance as a user who has the `ALTER USER` system
   privileges.
2. Grant permission for the IAM user to proxy to the local database user account.
   An IAM user cannot be referenced in the command so the proxy must be created
   between the database global user (mapped to the IAM user) and the target database
   user.
   In the following example, hrapp is the database schema to proxy to, and
   peterfitch_schema is the database global user exclusively mapped to user peterfitch.

   ```sql
   ALTER USER hrapp GRANT CONNECT THROUGH peterfitch_schema;
   ```

   At this stage, the IAM user can log in to the database instance using the proxy. For example,
to connect using a password verifier:

   ```sql
   CONNECT peterfitch[hrapp]@connect_string
   Enter password: password
   ```

   To connect using a token:

   ```sql
   CONNECT [hrapp]@connect_string
   ```

**Validate the IAM User Proxy Authentication**

You can validate the IAM user proxy configuration for both password and token authentication
methods.

1. Log in to the Autonomous Database instance as a user who has the `CREATE USER` and
   `ALTER USER` system privileges.
2. Connect at the IAM user and execute the `SHOW USER` and `SELECT SYS_CONTEXT`
   commands.
For example, suppose you want to check the proxy authentication of the IAM user `peterfitch` when they proxy to database user `hrapp`. You will need to connect to the database using the different types of authentication methods shown here, but the output of the commands that you execute will be the same for all types.

- For password authentication:

  ```sql
  CONNECT peterfitch/hrapp-password@connect_string
  SHOW USER;
  --The output should be USER is "HRAPP"
  SELECT SYS_CONTEXT('USERENV','AUTHENTICATION_METHOD') FROM DUAL;
  --The output should be "PASSWORD_GLOBAL"
  SELECT SYS_CONTEXT('USERENV','PROXY_USER') FROM DUAL;
  --The output should be "PETERFITCH_SCHEMA"
  SELECT SYS_CONTEXT('USERENV','CURRENT_USER') FROM DUAL;
  --The output should be "HRAPP"
  ```

- For token authentication:

  ```sql
  CONNECT [hrapp]/@connect_string
  SHOW USER;
  --The output should be USER is "HRAPP"
  SELECT SYS_CONTEXT('USERENV','AUTHENTICATION_METHOD') FROM DUAL;
  --The output should be "TOKEN_GLOBAL"
  SELECT SYS_CONTEXT('USERENV','PROXY_USER') FROM DUAL;
  --The output should be "PETERFITCH_SCHEMA"
  SELECT SYS_CONTEXT('USERENV','CURRENT_USER') FROM DUAL;
  --The output should be "HRAPP"
  ```

Disable Identity and Access Management (IAM) Authentication on Autonomous Database

Describes the steps to disable IAM external authentication user access for Autonomous Database.

You can disable IAM user access on your Autonomous Database instance as follows:

1. Run the `DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION` procedure.

   For example:

   ```sql
   BEGIN
     DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
   END;
   /
   ```

2. If you also want to update access to IAM from the resource, in this case the Autonomous Database instance, you may need to remove or modify the IAM group and the policies you set up to allow access to IAM from the Autonomous Database instance.
Notes for Using Autonomous Database Tools with Identity and Access Management (IAM) Authentication

Provides notes for using Autonomous Database tools with IAM authentication enabled.

- Oracle APEX is not supported for IAM users with Autonomous Database. See Create Oracle APEX Workspaces in Autonomous Database for information on using regular database users with Autonomous Database.
- Database Actions is not supported for IAM users with Autonomous Database. See Provide Database Actions Access to Database Users for information on using regular database users with Autonomous Database.
- Oracle Machine Learning Notebooks and other components are not supported for IAM Authorized users with Autonomous Database. See Add Existing Database User Account to Oracle Machine Learning Components for information on using regular database users with Autonomous Database.

Use Azure Active Directory (Azure AD) with Autonomous Database

You can configure an Autonomous Database instance for Azure AD users to connect using Azure OAuth2 access tokens.

Topics

- About Authorizing Microsoft Azure AD Users for an Oracle Autonomous Database
- Enable Azure AD Authentication on Autonomous Database
- Role and Schema Mapping for Azure AD Authentication on Autonomous Database
- Azure AD Client Configuration and Access for Autonomous Database

About Authorizing Microsoft Azure AD Users for an Oracle Autonomous Database

Users for Oracle Autonomous Database can be centrally managed in a Microsoft Azure Active Directory (Azure AD) service.

This type of integration enables the Azure AD user to access the Oracle Autonomous Database. Azure AD users and applications can log in with Azure AD Single Sign On (SSO) credentials to get an Azure AD OAuth2 access token to send to the database.

The administrator creates and configures the client application registration (app registration) of the Autonomous Database with Azure AD. The database administrator also creates application (app) roles for the database app registration in Azure AD, and assigns these roles to Azure AD users, groups, and applications. These app roles will be mapped to the database global schemas and global roles. An Azure AD principal that is assigned to an app role will be mapped to either a database global schema or database global role. An Oracle global schema can also be mapped exclusively to an Azure AD user. When the principal is a guest user or service principal, they can only be mapped to the database schema through an Azure app role. An Oracle global role can only be mapped to an Azure app role.
Oracle Autonomous Database tools including: Oracle APEX, Database Actions, Oracle Graph Studio, and Oracle Database API for MongoDB are not compatible with using Azure AD tokens to connect with the database.

Tools and applications that are updated to support Azure AD tokens can authenticate users directly with Azure AD and pass the database access token to the Autonomous Database instance. You can configure existing database tools such as SQL*Plus to use an Azure AD token from a file location. In these cases, Azure AD tokens can be retrieved using tools like Microsoft PowerShell or Azure CLI and put into a file location. An Azure AD OAuth2 database access token is a bearer token with an expiration time. The Oracle Database client driver will ensure that the token is in a valid format and that it has not expired before passing it to the database. The token is scoped for the database. Assigned app roles for the Azure AD principal are included as part of the access token. The directory location for the Azure AD token should only have enough permission for the user to write the token file to the location and the database client to retrieve these files (for example, just read and write by the process user). Because the token allows access to the database, it should be protected within the file system.

Azure AD users can request a token as a client registered with Azure AD app registration by using methods such as the following:

- Passing the Azure AD user name and password through a command line, script, file, or any other supported method
- Entering the Azure AD credentials into an Azure AD authentication screen with or without multi-factor authentication

Oracle Autonomous Database supports the following Azure AD authentication flows:

- Resource owner password credential (ROPC), which is used in non-graphic user interface environments when a pop-up window cannot be used to authenticate a user.
- Authorization code, which is used when a browser can be used to enter credentials for the user
- Client credentials, which are for applications that connect as themselves (and not the end-user)
- On-Behalf-Of (OBO), where an application requests an access token on behalf of a logged-in user to send to the database

Oracle Autonomous Database accepts tokens representing the following Azure AD principals:

- Azure AD user, who is registered user in the Azure AD tenancy
- Guest user, who is registered as a guest user in the Azure AD tenancy
- Service, which is the registered application connecting to the database as itself with the client credential flow (connection pool use case)

Architecture of the Microsoft Azure AD Integration with an Oracle Autonomous Database

Microsoft Azure Active Directory tokens follow the OAuth 2.0 standard with extensions. Using an Azure AD token to access an Oracle database is similar to using OCI IAM tokens.
The application or helper tool is responsible for requesting the access token from Azure AD. The token is passed to the database client through the API or by saving the token file in a directory location. Command-line tools such as Microsoft PowerShell or the Azure command-line interface can be used to retrieve the Azure AD token if the application cannot request the token.

The following diagram is a generalized flow diagram for OAuth 2.0 standard, using the OAuth2 token. See Authentication flow support in MSAL in the Microsoft Azure AD documentation for more details about each supported flow.

1. The Azure AD user requests access to the resource, the Oracle Autonomous Database instance.
2. The database client or application requests an authorization code from Azure AD.
3. Azure AD authenticates the Azure AD user and returns the authorization code.
4. The helper tool or application uses the authorization code with Azure AD to exchange it for the OAuth2 token.
5. The database client sends the OAuth2 access token to the Oracle Autonomous Database instance with the assigned app roles.
6. The Oracle Autonomous Database instance uses the Azure AD public key to verify that the access token was created by Azure AD.

Both the database client and the database server must be registered with the app registrations feature in the Azure Active Directory section of the Azure portal.

Azure AD Users Mapping to the Oracle Autonomous Database

Microsoft Azure users must be mapped to an Oracle Autonomous Database schema and have the necessary privileges (through roles) before being able to authenticate to the Oracle Autonomous Database instance.
In Microsoft Azure, users, groups, and applications can be assigned to the app roles by an Azure AD administrator.

An Azure AD user can be mapped to a database schema (user) either exclusively or through an app role.

- **Creating an exclusive mapping between an Azure AD user and an Oracle Database schema.** In this type of mapping, the database schema must be created for the Azure AD user with the appropriate database privileges and roles. The schema must also be updated during the life cycle of the Azure AD user and eventually dropped when the user leaves.

- **Creating a shared mapping between an Azure AD app role and an Oracle Database schema.** This type of mapping, which is more common than exclusive mappings, is for Azure AD users who have been assigned directly to the app role or is a member of an Azure AD group that is assigned to the app role. The app role is mapped to an Oracle Database schema (shared schema mapping). Shared schema mapping allows multiple Azure AD users to share the same Oracle Database schema so a new database schema is not required to be created every time a new user joins the organization. This operational efficiency allows database administrators to focus on database application maintenance, performance, and tuning tasks instead of configuring new users, updating privileges and roles, and removing accounts.

In addition to database roles and privileges being granted directly to the mapped global schema, additional roles and privileges can be granted through mapped global roles. Different Azure AD users mapped to the same shared global schema may need different privileges and roles. Azure app roles can be mapped to Oracle Database global roles. Azure AD users who are assigned to the app role or are a member of an Azure AD group that is assigned to the app role will be granted the Oracle Database global role when they access the database.

The following diagram illustrates the different types of assignments and mappings that are available.

These mappings are as follows:

- An Azure AD user can be mapped directly to an Oracle Autonomous Database global schema (user).
An Azure AD user, Azure AD group, or application is assigned to an app role, which is then mapped to either an Oracle Autonomous Database global schema (user) or a global role.

Use Cases for Connecting to an Oracle Autonomous Database Using Azure AD

Oracle Database supports three types of use cases for connecting to an Oracle Autonomous Database instance using Microsoft Azure Active Directory.

- **Connection using OAuth 2.0 authorization flow:** The client directs the resource owner to an authorization server, which in turn directs the resource owner back to the client with the authorization code. See the Microsoft Azure article [Microsoft identity platform and OAuth 2.0 authorization code flow](#).

- **Connection using the resource owner password credentials:** The resource owner password credentials (that is, the user name and password) can be used directly to obtain an access token. Azure AD requires an additional client Id and a secret for this flow. (The secret is not required for public client.) See the Microsoft Azure article [Microsoft identity platform and OAuth 2.0 Resource Owner Password Credentials](#).

- **Connection using the client credentials:** The client acts on its own behalf (the client is also the resource owner) or requests access to protected resources based on an authorization arranged with the authorization server. This flow is used to get the Azure OAuth2 access token for the service principal. An application can also request an Azure AD OAuth2 access token directly from Azure AD and pass it through a database client API. See the Microsoft Azure article [Get Azure AD tokens by using a service principal](#).

Enable Azure AD Authentication on Autonomous Database

An Azure AD administrator and an Autonomous Database administrator perform steps to configure Azure AD authentication on Autonomous Database.

**Topics**

- Registering the Oracle Autonomous Database Instance with a Microsoft Azure AD Tenancy
- Managing App Roles in Microsoft Azure AD
- Configuring Azure AD as an External Identity Provider for Autonomous Database

**Registering the Oracle Autonomous Database Instance with a Microsoft Azure AD Tenancy**

A user with administrator privileges uses Microsoft Azure AD to register the Oracle Autonomous Database instance with the Microsoft Azure AD tenancy.

1. Log in to the Azure portal as an administrator who has Microsoft Azure AD privileges to register applications.
2. In the Azure Active directory admin center page, from the left navigation bar, select **Azure Active Directory**.
3. In the MS - App registrations page, select **App registrations** from the left navigation bar.
4. Select **New registration**.

---

27-35
The Register an application window appears.

5. In the Register an application page, enter the following Oracle Autonomous Database instance registration information:
   - In the Name field, enter a name for the Autonomous Database instance connection (for example, Example Database).
   - Under Supported account types, select the account type that matches your use case.
     - Accounts in this organizational directory only (tenant_name only - Single tenant)
     - Accounts in any organizational directory (Any Azure AD directory - Multitenant)
     - Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
     - Personal Microsoft accounts only

6. Bypass the Redirect URI (Optional) settings. You do not need to create a redirect URI.

7. Click Register.

   After you click Register, Azure AD displays the app registration’s Overview pane, which will show the Application (client) ID under Essentials. This value is a unique identifier for the application in the Microsoft identity platform.

8. Register a scope, which will set the permission for the registered app.
   a. In the left navigation bar, select Expose an API.
b. Under Set the App ID URI, in the **Application ID URI** field, enter the app ID URI for the database connection using the following format, and then click **Save**:

\[ your\_tenancy\_url/application\_\{client\}\_id \]

In this specification:

- **your_tenancy_url** must include `https` as the prefix and the fully qualified domain name of your Azure AD tenancy.
- **application_{client}_id** is the ID that was generated when you registered the Oracle Autonomous Database instance with Azure AD. It is displayed in the Overview pane of the app registration.

For example:

`https://sales_west.example.com/1aa11111-1a1z-1a11-1a1a-11aa11a1aa1a`

c. Select **Add a scope** and then enter the following settings:
Add a scope

- **Scope name** specifies a name for the scope. Enter the following name:

  `session:scope:connect`

  This name can be any text. However, a scope name must be provided. You will need to use this scope name later when you give consent to the database client application to access the database.

- **Who can consent** specifies the necessary permissions. Select **Admins and users**, or for higher restrictions, **Admins only**.

- **Admin consent display name** describes the scope's purpose (for example, `Connect to Oracle`), which only administrators can see.

- **Admin consent display name** describes the scope's purpose (for example, `Connect to Example Database`), which only administrators can see.
• **User consent display name** is a short description of the purpose of the scope (for example, Connect to Example Database), which users can see if you specify **Admins and users** in **Who can consent**.

• **User consent description** is a more detailed description of the purpose of the scope (for example, Connect to Example Database), which users can see if you specify **Admins and users** in **Who can consent**.

• **State** enables or disables the connection. Select **Enabled**.

After you complete these steps, you are ready to add one or more Azure app roles, and then perform the mappings of Oracle schemas and roles.

**Related Topics**

- Quickstart: Register an application with the Microsoft identity platform

**Managing App Roles in Microsoft Azure AD**

In Azure AD, you can create and manage app roles that will be assigned to Azure AD users and groups and also be mapped to Oracle Database global schemas and roles.

**Creating a Microsoft Azure AD App Role**

Azure AD users, groups, and applications will be assigned to the app roles.

See the Microsoft Azure article [Create and assign a custom role in Azure Active Directory](#) for detailed steps on how to create an app role. The following steps describe how to create the app role for use with an Oracle Autonomous Database integration.

1. Log in to Azure AD as an administrator who has privileges for creating app roles.

2. Access the Oracle Database app registration that you created.
   a. Use the Directory + subscription filter to locate the Azure Active Directory tenant that contains the Oracle Database app registration.
   b. Select **Azure Active Directory**.
   c. Under **Manage**, select **App registrations**, and then select the Oracle Database instance that you registered earlier.

3. Under **Manage**, select **App roles**.

4. In the App roles page, select **Create app role**.

5. In the Create app role page, enter the following information:
   a. **Display name** is the displayed name of the role (for example, HR App Schema). You can include spaces in this name.
   b. **Value** is the actual name of the role (for example, HR_APP). Ensure that this setting matches exactly the string that is referenced in the application’s code. Do not include spaces in this name.
   c. **Description** provides a description of the purpose of this role.
   d. **Do you want to enable this app role?** enables you to activate the role.

6. Click **Apply**.

The app role appears in the App roles pane.
Assigning Users and Groups to the Microsoft Azure AD App Role

Before Microsoft Azure AD users can have access to the Oracle Autonomous Database instance, they must first be assigned to the app roles that will be mapped to Oracle Database schema users or roles.

See the Microsoft Azure article Add app roles to your application and receive them in the token for detailed steps assigning users and groups to an app role. The following steps explain how to do this for an Oracle Autonomous Database integration.

1. Log in to Azure AD as an administrator who has privileges for assigning Azure AD users and groups to app roles.

2. In Enterprise applications, access the Oracle Autonomous Database application that you registered.
   a. Use the Directory + subscription filter to locate the Azure Active Directory tenant that contains the Oracle connection.
   b. Select Azure Active Directory.
   c. Under Manage, select Enterprise applications, and then select the Oracle database application name that you registered earlier.

3. Under Getting Started, select Assign users and groups.

4. Select Add user/group.

5. In the Add assignment window, select Users and groups to display a list of users and security groups.

6. From this list, select the users and groups that you want to add to the app role, and then click Select.

7. In the Add assignment window, select Select a role to display a list of the app roles that you have created.

8. Select the app role and then select Select.

9. Click Assign.

Assigning an Application to an App Role

You can assign an Azure AD client application to an app role.

1. Log in to Azure AD as an administrator who has privileges for assigning Azure AD users and groups to app roles.
2. Access the app registration for the application.
3. Under Manage, select **API permissions**.
4. In the Configured permissions area, select + **Add a permission**.
5. In the Request API permission pane, select the **My APIs** tab.
6. Select the Oracle Database app that you want to give permission for this application to access. Then select the **Application permissions** option.
7. Select the database app roles to assign to the application and then click the **Add Permission** box at the bottom of the screen to assign the app roles and close the dialog box. Ensure that the app roles that you just assigned appear under Configured permissions.
8. Select **Grant admin consent for tenancy** to grant consent for the tenancy users, then select **Yes** in the confirmation dialog box.

**Related Topics**
- Configure the admin consent workflow

### Configuring Azure AD as an External Identity Provider for Autonomous Database

An Autonomous Database administrator can enable Azure AD as an external identity provider on an Autonomous Database instance. To enable Azure AD as an external identity provider:

1. Log in to the Autonomous Database instance as a user who has the **EXECUTE** privilege on the **DBMS_CLOUD_ADMIN** PL/SQL package. The **ADMIN** user has this privilege.
2. Run the **DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION** procedure with the Azure AD required parameters.

```sql
BEGIN
    DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
        type     => 'AZURE_AD',
        params   => JSON_OBJECT('tenant_id' => 'tenant_id',
                                'application_id' => 'application_id',
                                'application_id_uri' => 'application_id_uri'),
        force    => TRUE
    );
END;
```

27-41
In this procedure the Azure AD parameters are:

- **type**: Specifies the external authentication provider. For Azure AD, as shown, use 'AZURE_AD'.

- **params**: Values for the required Azure AD parameters are available from the Azure portal on the app registration Overview pane for Azure Active Directory. The required params for Azure AD are:
  
  - **tenant_id**: Tenant ID of the Azure Account. Tenant Id specifies the Autonomous Database instance’s Azure AD application registration.
  
  - **application_id**: Azure Application ID created in Azure AD to assign roles/schema mappings for external authentication in the Autonomous Database instance.
  
  - **application_id_uri**: Unique URI assigned to the Azure Application. This is the identifier for the Autonomous Database instance. The name must be domain qualified (this supports cross tenancy resource access). The maximum length for this parameter is 256 characters.

- **force**: Set this parameter to TRUE if another EXTERNAL AUTHENTICATION method is configured for the Autonomous Database instance and you want to disable it.

For example:

```
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type =>'AZURE_AD',
    params => JSON_OBJECT('tenant_id' VALUE '29981886-6fb3-44e3-82',
      'application_id' VALUE '11aa1a11-aaa',
      'application_id_uri' VALUE 'https://example.com/11a1a1aa'),
    force => TRUE
  );
END;
```

This sets the `IDENTITY_PROVIDER_TYPE` system parameter.

For example, you can use the following to verify `IDENTITY_PROVIDER_TYPE`:

```
SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME='identity_provider_type';
```

```
NAME                   VALUE
---------------------- --------
identity_provider_type AZURE_AD
```

See ENABLE_EXTERNAL_AUTHENTICATION Procedure for more information.
Role and Schema Mapping for Azure AD Authentication on Autonomous Database

Azure AD users are mapped to one database schema and optionally to one or more database roles. After mapping Azure AD users, user can connect to the Autonomous Database instance.

Topics

- Exclusively Mapping an Oracle Database Schema to a Microsoft Azure AD User
- Mapping a Shared Oracle Schema to an App Role
- Mapping an Oracle Database Global Role to an App Role

Exclusively Mapping an Oracle Database Schema to a Microsoft Azure AD User

You can exclusively map an Oracle Database schema to a Microsoft Azure AD user.

1. Log in to the Oracle Autonomous Database instance as a user who has been granted the CREATE USER or ALTER USER system privilege.
2. Run the CREATE USER or ALTER USER statement with the IDENTIFIED GLOBALLY AS clause specifying the Azure AD user name.
   For example, to create a new database schema user named peter_fitch and map this user to an existing Azure AD user named peter.fitch@example.com:

   ```sql
   CREATE USER peter_fitch IDENTIFIED GLOBALLY AS 'AZURE_USER=peter.fitch@example.com';
   ```
3. Grant the CREATE SESSION privilege to the user.
   ```sql
   GRANT CREATE SESSION TO peter_fitch;
   ```

Mapping a Shared Oracle Schema to an App Role

In this mapping, an Oracle schema is mapped to an app role.

1. Log in to the Oracle Autonomous Database instance as a user who has the CREATE USER or ALTER USER system privilege.
2. Run the CREATE USER or ALTER USER statement with the IDENTIFIED GLOBALLY AS clause specifying the Azure application role name.
   For example, to create a new database global user account (schema) named dba_azure and map it to an existing Azure AD application role named AZURE_DBA:

   ```sql
   CREATE USER dba_azure IDENTIFIED GLOBALLY AS 'AZURE_ROLE=AZURE_DBA';
   ```
Mapping an Oracle Database Global Role to an App Role

Oracle Database global roles that are mapped to Azure app roles give Azure users and applications additional privileges and roles above those that they have been granted through their login schemas.

1. Log in to the Oracle Autonomous Database instance as a user who has been granted the CREATE ROLE or ALTER ROLE system privilege.

2. Run the CREATE ROLE or ALTER ROLE statement with the IDENTIFIED GLOBALLY AS clause specifying the name of the Azure AD application role.

   For example, to create a new database global role named `widget_sales_role` and map it to an existing Azure AD application role named `WidgetManagerGroup`:

   ```sql
   CREATE ROLE widget_sales_role IDENTIFIED GLOBALLY AS 'AZURE_ROLE=WidgetManagerGroup';
   ```

Azure AD Client Configuration and Access for Autonomous Database

After you configure Azure AD on your Autonomous Database and you map Azure AD users, there are numerous ways that a user can configure a client to the Autonomous Database instance using Azure AD tokens.

See Configuring Azure AD Client Connections to the Oracle Database for more information.

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 a 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

You can configure Autonomous Database to authenticate and authorize Microsoft Active Directory users.

**Note:**

The CMU option supports Microsoft Active Directory servers but does not support the Azure Active Directory service. See Use Azure Active Directory (Azure AD) with Autonomous Database for information on using Azure Active Directory with 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 global users and global roles to Microsoft Active Directory users and groups.

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 AD DS Getting Started for more information.
- You must create an Oracle service directory user in Active Directory. See Connecting to Microsoft Active Directory for information on the Oracle service directory user account.
- An Active Directory system administrator must have installed Oracle password filter on the Active Directory servers, and set up Active Directory groups with Active Directory users to meet your requirements. 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 the CMU configuration database wallet, cwallet.sso and the CMU configuration file dsi.ora to configure CMU for your Autonomous Database. If you have configured CMU for an on-premise database, you can obtain these configuration files from your on-premise database server. If you have not configured CMU for an on-premise database, you need to create these files on your local computer, or on an on-premise database server. You can validate the wallet and the dsi.ora by configuring...
CMU for an on-premise database and verifying that an Active Directory user can successfully log on to the on-premise database with these configuration files. Then you upload these configuration files to the cloud in order to configure CMU for your Autonomous Database.

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 \texttt{dsi.ora} file for CMU, see Creating the \texttt{dsi.ora} File.

For details on configuring Active Directory for CMU and troubleshooting CMU for on-premise databases, see How To Configure Centrally Managed Users For Database Release 18c or Later Releases (Doc ID 2462012.1).

The following limitation applies to CMU with Active Directory on Autonomous Database:

- Only "password authentication" is supported for CMU with Autonomous Database. When you are using CMU authentication with Autonomous Database, other CMU authentication methods including Kerberos and PKI are not supported.

\textbf{Note:}

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

To configure Autonomous Database for CMU to connect to Active Directory servers:

1. Upload the CMU configuration files, including the database wallet file, \texttt{cwallet.sso} and the CMU configuration file, \texttt{dsi.ora} to your Object Store. This step depends on the Object Store you use.
   
The \texttt{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.

2. Run \texttt{DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION} procedure and pass in a location URI with the \texttt{params} JSON argument. You must place the configuration files \texttt{cwallet.sso} and \texttt{dsi.ora} in the Object Storage location specified in the \texttt{location_uri} parameter.

   For example:

   \begin{verbatim}
   BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
     type    => 'CMU',
     params  => JSON_OBJECT('location_uri' => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o',
                          'credential_name' => 'my_credential_name'));
   END;
   \end{verbatim}
Oracle recommends that you store the CMU configuration files in a private bucket in your Object Store.

In this example, `namespace-string` is the Oracle Cloud Infrastructure object storage namespace and `bucketname` is the bucket name. See Understanding Object Storage Namespaces for more information.

The `credential_name` you use in this step is the credentials to access the Object Store.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

If the `location_uri` is a pre-authenticated URL or a pre-signed URL, then supplying a `credential_name` is not required.

The procedure creates a directory object named `CMU_WALLET_DIR` in your database and copies the CMU configuration files from the Object Store location to the directory object. This procedure also sets the database property `CMU_WALLET` to the value 'CMU_WALLET_DIR' and sets the `LDAP_DIRECTORY_ACCESS` parameter value to the value `PASSWORD` to enable access from the Autonomous Database instance to Active Directory.

3. After you enable CMU authentication, remove the CMU configuration files including the database wallet `cwallet.sso` and the CMU configuration file `dsi.ora` from Object Store. You can use local Object Store methods to remove these files or use `DBMS_CLOUD.DELETE_OBJECT` to delete the files from Object Store.

```sql
Note:
See Disable Active Directory Access on Autonomous Database for instructions to disable the access from Autonomous Database to Active Directory.
```

See `ENABLE_EXTERNAL_AUTHENTICATION Procedure` for information on `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION`.

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

**Add Microsoft Active Directory Roles on Autonomous Database**

To add Active Directory roles, map the database global roles to Active Directory groups with `CREATE ROLE` or `ALTER ROLE` statements (and include the `IDENTIFIED GLOBALLY AS` clause).

To add global roles for Active Directory groups on Autonomous Database:

1. Log in as the ADMIN user to the database that is configured to use Active Directory (the ADMIN user has the `CREATE ROLE` and `ALTER ROLE` system privileges that you need for these steps).

2. Set the database authorization for Autonomous Database roles with `CREATE ROLE` or `ALTER ROLE` statement. Include the `IDENTIFIED GLOBALLY AS` clause and specify the DN of an Active Directory group.
Use the following syntax to map a directory user group to a database global role:

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

For example:

```
CREATE ROLE widget_sales_role IDENTIFIED GLOBALLY AS
  'CN=widget_sales_group,OU=sales,DC=production,DC=example,DC=com';
```

In this example all members of the `widget_sales_group` are authorized with the database role `widget_sales_role` when they log in to the database.

3. Use `GRANT` statements to grant the required privileges or other roles to the global role.

For example:

```
GRANT CREATE SESSION TO WIDGET_SALES_ROLE;
GRANT DWROLE TO WIDGET_SALES_ROLE;
```

`DWROLE` is a predefined role that has common privileges defined. See Manage User Privileges on Autonomous Database - Connecting with a Client Tool for information on setting common privileges for Autonomous Database users.

4. If you want to make an existing database role to be associated with an Active Directory group, then use `ALTER ROLE` statement to alter the existing database role to map the role to an Active Directory group.

Use the following syntax to alter an existing database role to map it to an Active Directory group:

```
ALTER ROLE existing_database_role
  IDENTIFIED GLOBALLY AS
  'DN_of_an_AD_GROUP_of_WHICH_the_AD_USER_IS_a_MEMBER';
```

5. If you want to create additional global role mappings for other Active Directory groups, follow these steps for each Active Directory group.

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

Add Microsoft Active Directory Users on Autonomous Database

To add Active Directory users to access a database, map database global users to Active Directory groups or users with `CREATE USER` or `ALTER USER` statements (with IDENTIFIED GLOBALLY AS clause).

The integration of Autonomous Database with Active Directory works by mapping Microsoft Active Directory users and groups directly to Oracle database global users and global roles.

To add global users for Active Directory groups or users on Autonomous Database:
1. Log in as the ADMIN user to the database that is configured to use Active Directory (the ADMIN user has the required `CREATE USER` and `ALTER USER` system privileges that you need for these steps).

2. Set database authorization for Autonomous Database users with `CREATE USER` or `ALTER USER` statements and include the `IDENTIFIED GLOBALLY AS` clause, specifying the DN of an Active Directory user or group.

   Use the following syntax to map a directory user to a database global user:

   ```sql
   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:

   ```sql
   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`:

   ```sql
   CREATE USER widget_sales IDENTIFIED GLOBALLY AS 'CN=widget_sales_group,OU=sales,DC=production,DC=example,DC=com';
   ```

   This creates a shared global user mapping. The mapping, with global user `widget_sales`, is effective for all users in the Active Directory group. Thus, anyone in the `widget_sales_group` can log in to the database using their Active Directory credentials (through the shared mapping of the `widget_sales` global user).

3. If you want Active Directory users to use an existing database user, own its schema, and own its existing data, then use `ALTER USER` to alter an existing database user to map the user to an Active Directory group or user.

   • Use the following syntax to alter an existing database user to map it to an Active Directory user:

     ```sql
     ALTER USER existing_database_user IDENTIFIED GLOBALLY AS 'DN_of_an_AD_USER';
     ```

   • Use the following syntax to alter an existing database user to map it to an Active Directory group:

     ```sql
     ALTER USER existing_database_user IDENTIFIED GLOBALLY AS 'DN_of_an_AD_GROUP_of_WHICH_the_AD_USER_IS_a_MEMBER';
     ```

4. If you want to create additional global user mappings for other Active Directory groups or users, follow these steps for each Active Directory group or user.

   See Configuring Authorization for Centrally Managed Users for more information on configuring users with Microsoft Active Directory.
Connect to Autonomous Database with Active Directory User Credentials

After the ADMIN user completes the CMU Active Directory configuration steps and creates global roles and global users, users log in to the database using their Active Directory username and password.

Note:
Do not log in using a Global User name. Global User names do not have a password and connecting with a Global User name will not be successful. You must have a global user mapping in your Autonomous Database in order to log in to the database. You cannot log in to the database with only global role mappings.

To log in to the database using an Active Directory username and password, connect as follows:

\[ \text{CONNECT "AD\_DOMAIN\"AD\_USERNAME"/AD\_USER\_PASSWORD@TNS\_ALIAS\_OF\_THE\_AUTONOMOUS\_DATABASE;} \]

For example:

\[ \text{CONNECT "production\pfitch"/password@adbname\_medium;} \]

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 \textit{pfitch in domain production}. The Active Directory user is a member of \textit{widget\_sales\_group} group which is identified by its DN
\[ 'CN=widget\_sales\_group,OU=sales,DC=production,DC=example,DC=com' \]

After configuring CMU with Active Directory on Autonomous Database and setting up Active Directory authorization, with global roles and global users, you can connect to your database using any of the connection methods described in Connecting to Autonomous Database. When you connect, if you want to use an Active Directory user then use Active Directory user credentials. For example, provide a username in this form, "\textit{AD\_DOMAIN\AD\_USERNAME}" (double quotes must be included), and use your \textit{AD\_USER\_PASSWORD} for the password.

If your Autonomous Database instance is in Restricted Mode, this mode only allows users with the \textit{RESTRICTED\_SESSION} privilege to connect to the database. The ADMIN user has this privilege. You can use restricted access mode to perform administrative tasks such as indexing, data loads, or other planned activities. See Change Autonomous Database Operation Mode to Read/Write Read-Only or Restricted for more information.
Tools Restrictions with Active Directory on Autonomous Database

Notes for using Autonomous Database tools with Active Directory:

- Oracle APEX is not supported for Active Directory users with Autonomous Database. See Create Oracle APEX Workspaces in Autonomous Database for information on using regular database users with Autonomous Database.
- Database Actions is not supported for Active Directory users with Autonomous Database. See Provide Database Actions Access to Database Users for information on using regular database users with Autonomous Database.
- Oracle Machine Learning Notebooks are not supported for Active Directory users with Autonomous Database. See Add Existing Database User Account to Oracle Machine Learning Components for information on using regular database users with Autonomous Database.

Verify Active Directory User Connection Information with Autonomous Database

When users log in to the database using their Active Directory username and password, you can verify and audit the user activity.

For example, when the user pfitch logs in:

```sql
CONNECT "production\pfitch"/password@exampleadb_medium;
```

The Active Directory user's log on username (samAccountName) is pfitch and widget_sales_group is the Active Directory Group name, and widget_sales is the database global user.

After pfitch logs in to the database, the command SHOW USER shows the global user name:

```sql
SHOW USER;
```

USER is "WIDGET_SALES"

The following command shows the DN (Distinguished Name) of the Active Directory user:

```sql
SELECT SYS_CONTEXT('USERENV', 'ENTERPRISE_IDENTITY') FROM DUAL;
```

For example you can verify this centrally managed user's enterprise identity:

```sql
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":

```sql
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
SQL> SELECT SYS_CONTEXT('USERENV', 'AUTHENTICATED_IDENTITY') FROM DUAL;
SYS_CONTEXT('USERENV','AUTHENTICATED_IDENTITY')
-----------------------------
production\pfitch
```

See Verifying the Centrally Managed User Logon Information for more information.

### Remove Active Directory Users and Roles on Autonomous Database

To remove Active Directory users and roles from Autonomous Databases, use standard database commands. This does not remove the related Active Directory users or groups that were mapped from the dropped database users or roles.

To remove users or roles from Autonomous Database:

1. Log in to the database that is configured to use Active Directory as a user who has been granted the `DROP USER` or `DROP ROLE` system privilege.
2. Drop the global users or the global roles that are mapped to Active Directory groups or users with `DROP USER` or `DROP ROLE` statement.

See Remove Users on Autonomous Database for more information.

### Disable Active Directory Access on Autonomous Database

Describes the steps to remove the CMU configuration from your Autonomous Database (and disable the LDAP access from your Autonomous Database to Active Directory).

After you configure your Autonomous Database instance to access CMU Active Directory, you can disable the access as follows:

- Use the `DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION` to disable CMU authentication. To run this procedure you must be logged in as `ADMIN` user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

For example:

```sql
BEGIN
    DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
END;
/
```

This disables CMU authentication on your Autonomous Database instance.

See DISABLE_EXTERNAL_AUTHENTICATION Procedure for more information.
Use Kerberos Authentication with Autonomous Database

Describes how to configure Kerberos to authenticate Oracle Autonomous Database users.

Topics
• About Kerberos Authentication
• Components of the Kerberos Authentication System
• Enable Kerberos Authentication on Autonomous Database
• Disable Kerberos Authentication on Autonomous Database
• Notes for Kerberos Authentication on Autonomous Database

About Kerberos Authentication

Kerberos is a strong network authentication protocol. It uses secret-key cryptography to enable strong authentication by providing user-to-server authentication.

• Oracle Autonomous Database support for Kerberos provides the benefits of single sign-on and centralized authentication of Oracle users. Kerberos is a trusted third-party authentication system that relies on shared secrets. It presumes that the third party is secure, and provides single sign-on capabilities, centralized password storage, database link authentication, and enhanced PC security. It does this through a Kerberos authentication server.

• The Kerberos system revolves around the concept of a ticket. A ticket is a set of electronic information that identifies a user or a service. A ticket identifies you and your network access privileges.

• In Kerberos-based authentication, you transparently send a request for a ticket to a Key Distribution Center (KDC). The Key Distribution Center authenticates you and grants you a ticket to access the database.

Components of the Kerberos Authentication System

Provides an overview of the Kerberos authentication system.

• A realm establishes an authentication administrative domain. Each realm has its own Kerberos database which contains the users and services for that particular administrative domain.

• Tickets are issued by the Key Distribution Center (KDC). Clients present tickets to the Database Server to demonstrate the authenticity of their identity. Each ticket has expiration and a renewal time.

• Keytabs stores long-term keys for one or more principals. A keytab file is generated by invoking the tool kadmin.local (for MIT Key Distribution Center) or ktpass (for Active Directory Key Distribution Center).

• Principals are the entries in the Key Distribution Center database. Each user, host or service is given a principal. A principal is a unique identity to which the Key Distribution Center can assign tickets.

• Kerberos support in Autonomous Database uses these values for various components that make up a service principal’s name:
To enable Kerberos authentication for your Autonomous Database, you must keep your Kerberos configuration files (krb.conf) and service key table file (v5srvtab) ready. For more information on these files and steps to obtain them, please see About Kerberos Configuration Files.

Enable Kerberos Authentication on Autonomous Database

Shows the steps to enable Kerberos authentication for Oracle Autonomous Database.

To run DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION you must be logged in as ADMIN user or have the EXECUTE privilege on DBMS_CLOUD_ADMIN.

To use DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION to enable Kerberos authentication:

1. Copy the Kerberos configuration files krb.conf and v5srvtab to a bucket in your Object Store.
2. Run DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION procedure and pass in a location URI with the params JSON argument. You must place the configuration files krb.conf and v5srvtab in the Object Storage location specified in the location_uri parameter.

For example:

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
      type => 'KERBEROS',
      params => JSON_OBJECT('location_uri' value 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
                         'credential_name' value 'my_credential_name')
   );
END;
/
```

**Note:**

Oracle recommends that you store the Kerberos configuration files in a private bucket in your Object Store.
In this example, `namespace-string` is the Oracle Cloud Infrastructure object storage namespace and `bucketname` is the bucket name. See Understanding Object Storage Namespaces for more information.

The `credential_name` you use in this step is the credentials for the Object Store.

Creating a credential to access Oracle Cloud Infrastructure Object Store is not required if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

If the `location_uri` is a pre-authenticated URL then supplying a `credential_name` is not required.

This creates a directory object named `KERBEROS_DIR` in your database and uses the credential to download the Kerberos configuration files from the Object Store location to the directory object.

3. After you enable Kerberos authentication, remove the configuration `krb.conf` and `v5srvtab` from Object Store. You can use local Object Store methods to remove these files or use `DBMS_CLOUD.DELETE_OBJECT` to delete the files from Object Store.

See Navigate to Oracle Cloud Infrastructure Object Storage and Create Bucket for more information on Object Storage.

See Upload Files to Your Oracle Cloud Infrastructure Object Store Bucket for more information on uploading files to Object Storage.

See ENABLE_EXTERNAL_AUTHENTICATION Procedure for more information.

See Configuring Kerberos Authentication for more information on keytab files.

Disable Kerberos Authentication on Autonomous Database

Shows the steps to disable Kerberos authentication for your Autonomous Database instance.

- Run `DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION` procedure to disable Kerberos authentication. To run the procedure, you must be logged in as ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

```sql
BEGIN
    DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
END;
/
```

This disables the Kerberos authentication (or any external authentication scheme specified) for Oracle Autonomous Database.

See DISABLE_EXTERNAL_AUTHENTICATION Procedure for more information.

Notes for Kerberos Authentication on Autonomous Database

Provides the notes on using Kerberos Authentication for Autonomous Database.

- If you enable Kerberos authentication for your Autonomous Database, you can still use password-based database authentication for your database. However, you can use only one external authentication method at a time.

- Kerberos authentication is not supported for these tools:
You can enable Kerberos authentication to authenticate the ADMIN user. You can use the Reset Password functionality on the Oracle Cloud Infrastructure Console to reset the ADMIN user's password and regain access if a corrupted keytab file causes ADMIN user's authentication to fail.

The default value for the maximum clock skew in Autonomous Database is 300 seconds (5 minutes). You cannot change the default clock skew value.
Managing and Monitoring Performance of Autonomous Database

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

Topics

• Monitor the Performance of Autonomous Database
• Manage Concurrency and Priorities on Autonomous Database
• Manage CPU/IO Shares on Autonomous Database
• Manage Runaway SQL Statements on Autonomous Database
• Manage Optimizer Statistics on Autonomous Database
• Manage Automatic Indexing on Autonomous Database
• Manage Automatic Partitioning on Autonomous Database
• Monitor Autonomous Database with Performance Hub
• Monitor the Performance of Autonomous Database with Oracle Management Cloud
• Use Database Management Service to Monitor Databases
• Monitor Performance with Autonomous Database Metrics
• Perform SQL Tracing on Autonomous Database
• Monitor Autonomous Database Availability
• Monitor Regional Availability of Autonomous Databases
• Service Console Replacement with Database Actions
• Available Metrics: oci_autonomous_database

Monitor the Performance of Autonomous Database

The Overview and Activity tabs on the Monitor Database card in Database Actions provide information about the performance of an Autonomous Database instance. The Activity tab shows past and current monitored SQL statements and detailed information about each statement.

Topics

• Use Database Actions to Monitor Activity and Utilization
• Use Database Actions to Monitor Active Session History Analytics and SQL Statements
Use Database Actions to Monitor Activity and Utilization

Database Actions provides the Database Monitor card, with **Overview** and **Monitor** tabs to provide real-time and historical information about the utilization of an Autonomous Database instance.

Perform the following prerequisite steps as necessary:

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

To view the **Overview** tab that shows general information about utilization, do the following:

1. Access Database Actions as the ADMIN user.
   
   See [Access Database Actions as ADMIN](#) for more information.

2. On the Database Actions Launchpad, under Monitoring, click **Database Monitor**.

   ![Note](Note.png)

   You can bookmark the Launchpad URL and go to that URL directly without logging in to the Oracle Cloud Infrastructure Console. If you logout and use the bookmark, then you need to enter the **ADMIN** username and **password**, and click **Sign in**. See [Set the ADMIN Password in Autonomous Database](#) if you need to change the password for the **ADMIN** user.

Database Monitor Overview

The **Overview** tab shows real-time and historical information about the Autonomous Database utilization.

The charts shown on this page include:

- **Storage**: This chart shows the provisioned, allocated, and used storage. The chart indicates what percentage of the space is currently in-use.
Provisioned storage is the amount of storage you select when you provision the instance or when you modify storage by scaling storage.

**Storage allocated** is the amount of storage physically allocated to all data tablespaces and temporary tablespaces and includes the free space in these tablespaces. This does not include storage for the sample schemas.

**Storage used** is the amount of storage actually used in all data and temporary tablespaces. This does not include storage for the sample schemas. The storage used is the storage in the Autonomous Database as follows:

- Storage used by all database objects. Note: the chart does not include storage for the sample schemas as they do not count against your storage.
- Storage for files users put in the file system.
- Storage used by temporary tablespaces.
- Used storage excludes the free space in the data and temporary tablespaces.

By default the chart does not show the used storage. Select **Storage used** to expand the chart to see used storage (the values are calculated when you open the chart).

For an Autonomous JSON Database the chart shows an additional field showing the percentage of storage used that is not storing JSON documents.

---

**Note:**

If you drop an object, the space continues to be consumed until you empty the recycle bin. See Purging Objects in the Recycle Bin for more information.
See Use Sample Data Sets in Autonomous Database for information on sample schemas SH and SSB.

- **CPU utilization (%)**: This chart shows the historical CPU utilization of the service:
  - OCPU 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.

  - OCPU auto scaling enabled: For databases with OCPU 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**

Notes for display results:

- **OCPU auto scaling disabled**: For databases with OCPU 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.

- **OCPU auto scaling enabled**: For databases with OCPU auto scaling enabled, for each hour the chart shows the average number of OCPUs used during that hour if that value is higher than the number of OCPUs provisioned. If the number of OCPUs used is not higher than the number of OCPUs provisioned, then the chart shows the number of OCPUs allocated for that hour.

- **Stopped Database**: If the database was stopped for the full hour the chart shows 0 OCPUs allocated for that hour.

Click **Show details** for more information, including the number of OCPUs allocated to the database and to external resources, and the total allocated OCPUs.

The Show details view includes separate values for database OCPU usage and external resource OCPU usage. External resources include: Cloud SQL, Graph, OML4PY, and others. The Total OCPUs are the total number of OCPUs in use on the Autonomous Database. The external OCPUs value shows how external OCPUs contribute to the total OCPU usage.

• **SQL statement response time (s)**: This chart shows the average response time, in seconds, of SQL statements historically. This chart shows hourly data. A data point shows the average SQL statement response time for that hour. For example, a data point

---

**Chapter 28**

Monitor the Performance of Autonomous Database

28-5
at 10:00 shows the average SQL statement response time, in seconds, for the hour from 9:00-10:00.

- **SQL statements executed per second**

  ![SQL statement response time chart](image)

  **Note:**
  Database Monitor does not show this chart when the Autonomous Database instance workload type is **Data Warehouse**.

The default retention period for performance data is thirty (30) days. The CPU utilization, running statements, and average SQL response time charts show data for the last eight (8) days by default.
Note:
You can change the retention period by modifying the Automatic Workload Repository retention setting with the PL/SQL procedure `DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS()`. Be aware that increasing the retention period will result in more storage usage for performance data. See Oracle Database PL/SQL Packages and Types Reference.

Database Monitor Activity

The Monitor tab shows real-time and historical information about the Autonomous Database performance data, activity, and utilization.

Note:
The default view in the Monitor tab is real-time. This view shows performance data for the last hour.

The charts on this page are:

- **Database Activity**
  This chart shows the average number of sessions in the database using CPU or waiting on a wait event. See Oracle Database Reference for more information on wait events.

- **CPU Utilization**
  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 OCPU 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 Manage Concurrency and Priorities on Autonomous Database for detailed information on consumer groups.

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

- **Queued Statements**
  This chart shows the average number of queued SQL statements in each consumer group.
  
  See Manage Concurrency and Priorities on Autonomous Database for detailed information on consumer groups.
To see earlier data click **Time period**. The default retention period for performance data is thirty (30) days. By default in the Time Period view the charts show information for the last eight (8) days.

In the time period view you can use the calendar to look at a specific time period in the past 30 days. You can also use the time slider to change the period for which performance data is shown.

**Note:**

The retention time can be changed by changing the Automatic Workload Repository retention setting with the PL/SQL procedure `DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS`. Be aware that increasing the retention time results in more storage usage for performance data. See [Oracle Database PL/SQL Packages and Types Reference](https://docs.oracle.com/en/database/oracle/plsql/plsql-ref.html).

### Use Database Actions to Monitor Active Session History Analytics and SQL Statements

In Database Actions the Performance Hub card provides information about Active Session History (ASH) analytics and current and past monitored SQL statements.

Perform the following prerequisite step as necessary:

- Access Database Actions as the ADMIN user. See Access Database Actions as ADMIN for more information.

### The Performance Hub Page

The Performance Hub page shows performance data for a time period you specify. To navigate to the Performance Hub page, do either of the following:

- In the Launchpad page, click **Performance Hub**.
- Click **Selector** to display the navigation menu. Under Monitoring, select **Performance Hub**.

**Note:**

The Performance Hub page is available in the following user interface languages: French, Japanese, Korean, Traditional Chinese, and Simplified Chinese. If you change the language to German, Spanish, Italian, or Portuguese in Preferences, the Performance Hub page reverts to English.

The Performance Hub page consists of these parts:

- **Time Range Area**: Use the controls in time range area at the top of the page to specify the time period for which you want to view performance data.
- **ASH Analytics Tab**: Use this tab to explore ASH (Active Session History) information across a variety of different dimensions for the specified time period.
• **SQL Monitoring Tab:** Use this tab to view the top 100 SQL statement executions by different dimensions for the specified time period, and to view details of SQL statement executions you select.

**Time Range Area**

Use these controls in the time range area to specify the time period for which you want to view performance data:

- **Select Duration:** Use this drop-down list, located in the top right of the time range area, to set the timeframe displayed in the timeline. You can choose **Last hour, Last 8 hours, Last 24 hours, Last Week,** or you can choose **Custom** and define your own timeframe.

- **Timeline:** The timeline displays a graph spanning the timeframe selected in the timeframe dropdown, showing Waits, User I/O and CPU usage during the period. At its end is the time slider.

- **Time Slider:** The time slider is a box you can drag back and forth along the current timeline. Use it to pick the specific time period within the timeframe for which you want to view performance data. You can also drag the side handles on the time slider to make it wider or narrower to encompass a longer or shorter time period.

**ASH Analytics Tab**

The ASH Analytics tab consists of the Average Active Sessions chart and two secondary tables below it.

- **Average Active Sessions Chart:** This chart shows performance information for the time period defined by the time slider. You can choose to chart different dimensions of information, such as **Wait Class, Wait Event,** or **Service,** by selecting the dimension from the drop-down list to the right of the chart title.

You can download an AWR (Automatic Workload Repository) report for the current time period by right-clicking in the Average Active Sessions chart area and choosing **Generate AWR Report.**

For more information about ASH and AWR, see Active Session History (ASH) and Automatic Workload Repository (AWR) in *Oracle Database Concepts.*

- **Secondary Tables:** The two tables below the Average Active Sessions chart show the information dimension chosen in the chart filtered by another dimension you choose. For example, if the Average Active Sessions chart is showing Wait Class, you could show SQL ID and User Session dimension information by Wait Class, one in each of the two secondary tables.

  In the secondary tables, the SQL ID and User Session dimensions provide links to SQL Details and Session Details pages, respectively, for the dimension data listed in the table.

**SQL Monitoring Tab**

The SQL Monitoring tab shows a table of the top 100 monitored SQL statements that were executing or that completed during the selected time period.

The table displays information about monitored SQL statement executions. If there is a green spinning icon in the Status column, then the monitored statement did not complete during the selected time period. A red cross indicates that the SQL did not complete either due to an error or due to the session getting terminated. If there is a check mark in the Status column, then the statement completed its execution during the selected time period.
SQL statements are monitored only if they have consumed at least 5 seconds of CPU or I/O time.

You can view information such as the status of a statement, its duration, its type (SQL, PL/SQL, or DBOP), its SQL ID, its SQL plan hash, the user who issued it, whether it executed as a serial or parallel statement, the time the database spent performing CPU activity, I/O, or other activity for the statement, the read and write requests and bytes associated with the statement, and the start and end time for the statement.

Click a SQL ID to display the SQL Details page with more information about that SQL statement.

Manage Concurrency and Priorities on Autonomous Database

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

Topics
- Database Service Names for Autonomous Data Warehouse
- Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database
- Idle Time Limits
- Service Concurrency
- Change MEDIUM Service Concurrency Limit
- Predefined Job Classes with Oracle Scheduler

Database Service Names for Autonomous Data Warehouse

You are required to select a service when you connect to the database. The service names for Autonomous Data Warehouse connections are in the format:

- databasename_high
- databasename_medium
- databasename_low

These services map to the LOW, MEDIUM, and HIGH consumer groups.

For example, if you create an Autonomous Database with a Data Warehouse workload type and specify the database name as DB2020, your service names are:

- db2020_high
- db2020_medium
- db2020_low

If you connect using the db2020_low service, the connection uses the LOW consumer group.

The basic characteristics of these consumer groups are:

- HIGH: Highest resources, lowest concurrency. Queries run in parallel.
• MEDIUM: Less resources, higher concurrency. Queries run in parallel.

Picking one of the predefined services provides concurrency values that work well for most applications. In cases where selecting one of the default services does not meet your application’s performance needs, you can use the MEDIUM service and modify the concurrency limit. For example, when you run single-user benchmarks, you can set the concurrency limit of the MEDIUM service to 1 in order to obtain the highest degree of parallelism (DOP).

See Change MEDIUM Service Concurrency Limit for more information.

• LOW: Least resources, highest concurrency. Queries run serially.

Note:

After connecting to the database using one service, do not attempt to manually switch that connection to a different service by simply changing the consumer group of the connection. When you connect using a service, Autonomous Database performs more actions to configure the connection than just setting its consumer group.

Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database

You are required to select a service when you connect to the database. The service names for connecting to Autonomous Transaction Processing or Autonomous JSON Database are in the format:

• `databasename_tpurgent`
• `databasename_tp`
• `databasename_high`
• `databasename_medium`
• `databasename_low`

These services map to the `TPURGENT`, `TP`, `HIGH`, `MEDIUM` and `LOW` consumer groups.

For example, if you create an Autonomous Database with a Transaction Processing workload type and specify the database name as `DB2020`, your connection service names are:

• `db2020_tpurgent`
• `db2020_tp`
• `db2020_high`
• `db2020_medium`
• `db2020_low`

If you connect using the `db2020_tp` service, the connection uses the `TP` consumer group.

The basic characteristics of these consumer groups are:

• `TPURGENT`: The highest priority application connection service for time critical transaction processing operations. This connection service supports manual parallelism.
• TP: A typical application connection service for transaction processing operations. This connection service does not run with parallelism.

• HIGH: A high priority application connection service for reporting and batch operations. All operations run in parallel and are subject to queuing.

• MEDIUM: A typical application connection service for reporting and batch operations. All operations run in parallel and are subject to queuing.

Picking one of the predefined services provides concurrency values that work well for most applications. In cases where selecting one of the default services does not meet your application’s performance needs, you can use the MEDIUM service and modify the concurrency limit. For example, when you run single-user benchmarks, you can set the concurrency limit of the MEDIUM service to 1 in order to obtain the highest degree of parallelism (DOP).

See Change MEDIUM Service Concurrency Limit for more information.

• LOW: A lowest priority application connection service for reporting or batch processing operations. This connection service does not run with parallelism.

Note:

After connecting to the database using one service, do not attempt to manually switch that connection to a different service by simply changing the consumer group of the connection. When you connect using a service, Autonomous Database performs more actions to configure the connection than just setting its consumer group.

Idle Time Limits

Autonomous Database has predefined idle time limits for sessions so that idle sessions do not hold system resources for a long time.

A session may be terminated if it stays idle for more than five (5) minutes and the resources it consumes are needed by other users. This allows other active sessions to proceed without waiting for the idle session.

If you want sessions to be terminated after a certain amount of time, independent of the consumed resources needed by other users, then set the MAX_IDLE_TIME initialization parameter. The MAX_IDLE_TIME parameter specifies the maximum number of minutes that a session can be idle. After the specified amount of time, MAX_IDLE_TIME kills sessions.

Note:

Sessions that are idle for more than 48 hours are terminated whether they are holding resources or not.

See MAX_IDLE_TIME for more information.
Service Concurrency

The consumer groups of the predefined service names provide different levels of performance and concurrency. The available service names are different depending on your workload: Data Warehouse, Transaction Processing, or JSON Database.

Picking one of the predefined services provides concurrency values that work well for most applications. In cases where selecting one of the default services does not meet your application's performance needs, you can use the MEDIUM service and modify the concurrency limit. For example, when you run single-user benchmarks, you can set the concurrency limit of the MEDIUM service to 1 in order to obtain the highest degree of parallelism (DOP).

See Change MEDIUM Service Concurrency Limit for more information.

In this topic, the "number of OCPUs" is the OCPU count shown in the Oracle Cloud Infrastructure Console.

Service Concurrency Limits for Data Warehouse Workloads

The tnsnames.ora file provided with the credentials zip file contains three database service names identifiable as high, medium and low for Autonomous Database with Data Warehouse workloads.

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

<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>$1.26 \times \text{number of OCPUs}$</td>
</tr>
<tr>
<td>low</td>
<td>$300 \times \text{number of OCPUs}$</td>
</tr>
</tbody>
</table>

For example, for an Autonomous Database 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 the MEDIUM and HIGH consumer groups, new SQL statements in that consumer group will be queued until one or more running statements finish. With the LOW consumer group, when the concurrency limit is reached you will not be able to connect new sessions.

The following table shows sample concurrent connections 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 4800</td>
</tr>
</tbody>
</table>
Service Concurrency Limits for Transaction Processing and JSON Database Workloads

The `tnsnames.ora` file provided with the credentials zip file contains five database service names identifiable as `tpurgent`, `tp`, `high`, `medium`, and `low` for Autonomous Database with Transaction Processing or JSON Database workloads.

The following shows the details for the default number of concurrent statements for each connection service for Transaction Processing or JSON Database workloads.

<table>
<thead>
<tr>
<th>Database Service Name</th>
<th>Concurrent Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tpurgent</code></td>
<td>300 × number of OCPUs</td>
</tr>
<tr>
<td><code>tp</code></td>
<td>300 × number of OCPUs</td>
</tr>
<tr>
<td><code>high</code></td>
<td>3</td>
</tr>
<tr>
<td><code>medium</code></td>
<td>1.26 × number of OCPUs</td>
</tr>
<tr>
<td><code>low</code></td>
<td>300 × number of OCPUs</td>
</tr>
</tbody>
</table>

Change MEDIUM Service Concurrency Limit

If your application requires customized concurrency, you can modify the concurrency limit for your Autonomous Database MEDIUM service.

Picking one of the predefined services provides concurrency values that work well for most applications. In cases where selecting one of the default services does not meet your application’s performance needs, you can use the MEDIUM service and modify the concurrency limit. For example, when you run single-user benchmarks, you can set the concurrency limit of the MEDIUM service to 1 in order to obtain the highest degree of parallelism (DOP).

**Note:**

Changing the concurrency limit is only allowed for an instance that has two (2) or more OCPUs.

For example, if your instance is configured with 100 OCPUs, by default Autonomous Database provides a concurrency limit of 126 for the MEDIUM service:

1.26 x number of OCPUs sessions (up to 126 concurrent queries)

Thus, in this example using the MEDIUM service supports an application with up to 126 concurrent queries with DOP of 4. If you only need 50 concurrent queries and you want a higher DOP you can decrease the concurrency limit and thus increase the DOP. To do this, set the MEDIUM service concurrency limit to 50. When you change the concurrency limit the system calculates and sets the DOP based on the concurrency limit you select and the number of OCPUs. For this example, with the concurrency limit set to 50, the new DOP is 12.

You can change the concurrency limit for the MEDIUM service in Database Actions or using the PL/SQL package `CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE`. 
Follow these steps to change the MEDIUM service concurrency limit in Database Actions:

1. Access Database Actions as the ADMIN user.
   See Access Database Actions as ADMIN for more information.


   ![Set Resource Management Rules](image)

   - **Run-away criteria**
     - CPU/MIO shares
     - Concurrency limit
     - Degree of parallelism (DOP)
   - **Consumer group**
     - HIGH
       - Concurrency limit: 3
       - Degree of parallelism (DOP): 6
     - MEDIUM
       - Concurrency limit: 4
     - LOW
       - Concurrency limit: 1800
   - Your OCPU count is 6 and Auto Scaling is enabled for your instance.

4. For the MEDIUM service, change the value to the desired concurrency limit by entering a value or by clicking the Decrement or Increment icons.
   If the concurrency limit you specify is not valid, based on the number of OCPUs, you will receive a message such as the following, listing the valid range of values for your instance:

   **Please enter a concurrency limit between 1 and 300**

   This error message example is from an instance with 100 OCPUs (the 300 maximum value shown is 3 x number of OCPUs).

5. Click Save Changes.

6. Click OK.
   To reset the concurrency limit for the MEDIUM service to its default value, click Load Default Values and click Save Changes.
Change MEDIUM Service Concurrency Limit with PL/SQL Procedure

UPDATE_PLAN_DIRECTIVE

As an alternative to using the Set Resource Management Rules card in Database Actions, you can use PL/SQL to change the concurrency limit for the MEDIUM service.

To change the MEDIUM service concurrency limit with CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE:

1. Call the PL/SQL procedure CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE to update the concurrency limit for the MEDIUM consumer group.

   For example, with 3 OCPUs, change the MEDIUM service’s concurrency limit to 2, as follows:

   BEGIN
   -- Call the PL/SQL procedure to update the concurrency limit
   CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'MEDIUM', concurrency_limit => 2);
   END;
   /

   If the concurrency_limit you specify is not valid, based on the number of OCPUs, you will receive a message such as the following, listing the valid range of values for your instance:

   ORA-20000: Invalid or missing value. Concurrency limit must be between 1 and 9 for the specified OCPU count

   This error message example is from an instance with 3 OCPUs.

2. Use the PL/SQL function CS_RESOURCE_MANAGER.LIST_CURRENT_RULES to verify the updated MEDIUM service concurrency limit and degree of parallelism:

   SELECT * FROM CS_RESOURCE_MANAGER.LIST_CURRENT_RULES();

   CONSUMER_GROUP ELAPSED_TIME_LIMIT IO_MEGABYTES_LIMIT SHARES CONCURRENCY_LIMIT DEGREE_OF_PARALLELISM
   ----------------- ------------------ ------------------ ------ -----
   HIGH              3                 3
   MEDIUM            2                 2
   LOW               900                9

   This procedure returns the list of values for all consumer groups. After you modify the concurrency limit as specified in Step 1, check the MEDIUM service CONCURRENCY_LIMIT and DEGREE_OF_PARALLELISM values to verify your changes.
3. After you change the concurrency limit for the MEDIUM service, test your application by connecting with the MEDIUM service to verify that the customized concurrency limit meets your performance objectives.

When you want to go back to the default values, use the `CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES` PL/SQL procedure to revert to the default settings for the MEDIUM service.

For example:

```sql
BEGIN
    CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'MEDIUM',
                                                concurrency_limit => TRUE);
END;
/
```

See `CS_RESOURCE_MANAGER Package` for more information.

### Change MEDIUM Service Concurrency Limit Notes

- Changing the concurrency limit is only allowed for the MEDIUM service.
- Changing the concurrency limit is only allowed when the number of OCPUs is greater than 1.
- Changing the concurrency limit also changes the degree of parallelism (in most cases, depending on the magnitude of the difference between the old concurrency limit and the new value you set).
- The concurrency limit you set must be in the range between: 1 and 3 x the number of OCPUs.
- The MEDIUM service sets the following concurrency limit and DOP values by default:

<table>
<thead>
<tr>
<th>MEDIUM Database Service</th>
<th>Default Value</th>
</tr>
</thead>
</table>
| Concurrency Limit       | 1.26 × number of OCPUs when the number of OCPUs ≥ 4  
                          | 5 when the number of OCPUs < 4 |
| DOP                     | 4 when the number of OCPUs 4  
                          | or  
                          | The number of OCPUs, when the number of OCPUs < 4 |

- By changing the value of the concurrency limit, the DOP of the MEDIUM service can go as low as 2 and as high as: 2 x number of OCPUs (if OCPU auto scaling is disabled) or 6 x number of OCPUs (if OCPU auto scaling is enabled).
- At any time you can return to the default values for the MEDIUM service concurrency limit and DOP.

### Predefined Job Classes with Oracle Scheduler

Autonomous Database includes predefined `job_class` values to use with Oracle Scheduler.

The predefined `job_class` values, TPURGENT, TP, HIGH, MEDIUM and LOW map to the corresponding consumer groups. These job classes allow you to specify the consumer group a job runs in with `DBMS_SCHEDULER.CREATE_JOB`.
The `DBMS_SCHEDULER.CREATE_JOB` procedure supports PLSQL_BLOCK and STORED_PROCEDURE job types for the `job_type` parameter in Autonomous Database.

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

```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;
/
```

Notes for Oracle Scheduler:

- To use `DBMS_SCHEDULER.CREATE_JOB` additional grants for specific roles or privileges might be required. The `ADMIN` user and users with `DWROLE` have the required `CREATE SESSION` and `CREATE JOB` privileges. If a user does not have `DWROLE` then grants are required for `CREATE SESSION` and `CREATE JOB` privileges.

- The `instance_id` job attribute is ignored for Oracle Scheduler jobs running on Autonomous Database.

See Scheduling Jobs with Oracle Scheduler for more information on Oracle Scheduler and `DBMS_SCHEDULER.CREATE_JOB`.

See SET_ATTRIBUTE Procedure for information on job attributes.

**Manage CPU/IO Shares on Autonomous Database**

Autonomous Database comes with predefined CPU/IO shares assigned to different consumer groups. You can modify these predefined CPU/IO shares if your workload requires different CPU/IO resource allocations.

The CPU/IO shares assigned to the consumer groups determine the CPU/IO resources a consumer group can use with respect to the other consumer groups. The default CPU/IO shares depend on the Autonomous Database workload.

<table>
<thead>
<tr>
<th>Workload Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Warehouse</td>
<td>By default, the CPU/IO shares assigned to the consumer groups HIGH, MEDIUM, LOW are 4, 2, and 1, respectively. With the default settings the consumer group HIGH will be able to use 4 times more CPU/IO resources compared to LOW and 2 times more CPU/IO resources compared to MEDIUM, when needed. The consumer group MEDIUM will be able to use 2 times more CPU/IO resources compared to LOW, when needed.</td>
</tr>
</tbody>
</table>
Workload Type | Details
---|---
Transaction Processing | By default, the CPU/IO shares assigned to the consumer groups TPURGENT, TP, HIGH, MEDIUM, and LOW are 12, 8, 4, 2, and 1, respectively. With the default settings the consumer group TPURGENT will be able to use 12 times more CPU/IO resources compared to LOW, when needed. The consumer group TP will be able to use 4 times more CPU/IO resources compared to MEDIUM, when needed.

You can set CPU/IO shares in Database Actions or using the PL/SQL package CSRESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE.

To use Database Actions to change the CPU/IO share values for consumer groups:

1. Access Database Actions as the ADMIN user.  
   See Access Database Actions as ADMIN for more information.
3. Select the CPU/IO shares tab to set CPU/IO share values for consumer groups.
4. Set the desired CPU/IO share value for a consumer group by entering a value or by clicking the Decrement or increment icons.
5. Click Save Changes.
6. Click OK.

To reset CPU/IO shares values to the defaults, click Load Default Values and click Save Changes to apply the populated values.

As an alternative to using Database Actions, you can use the PL/SQL procedure CSRESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE to change the CPU/IO share values for consumer groups.

For example, on an Autonomous Data Warehouse database, run the following script as the ADMIN user to set the CPU/IO shares to 8, 2, and 1 for consumer groups HIGH, MEDIUM, and LOW respectively. This allows the consumer group HIGH to use 4 times more CPU/IO resources compared to the consumer group MEDIUM and 8 times CPU/IO resources compared to the consumer group LOW:

```sql
BEGIN
  CSRESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'HIGH', shares => 8);
  CSRESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'MEDIUM', shares => 2);
  CSRESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'LOW', shares => 1);
END;
/```

For example, on an Autonomous JSON Database or on an Autonomous Transaction Processing database, run the following script as the ADMIN user to set CPU/IO shares to 12, 4, 2, 1, and 1 for the consumer groups TPURGENT, TP, HIGH, MEDIUM, and LOW respectively. This allows the consumer group TPURGENT to use 3 times more CPU/IO resources compared to LOW, when needed.
resources compared to the consumer group TP and 12 times CPU/IO resources compared to the consumer group MEDIUM:

BEGIN
  CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'TPURGENT', shares => 12);
  CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'TP', shares => 4);
  CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'HIGH', shares => 2);
  CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'MEDIUM', shares => 1);
  CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'LOW', shares => 1);
END;
/

When you want to go back to the default shares values you can use the PL/SQL procedure CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES to revert to the default settings.

For example, on an Autonomous Data Warehouse database, run the following script as the ADMIN user to set the CPU/IO shares to default values for the consumer groups HIGH, MEDIUM, and LOW:

BEGIN
  CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'HIGH', shares => TRUE);
  CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'MEDIUM', shares => TRUE);
  CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'LOW', shares => TRUE);
END;
/

For example, on an or on an Autonomous JSON Database or on an Autonomous Transaction Processing database, run the following script as the ADMIN user to set the default values for CPU/IO shares for the consumer groups TPURGENT, TP, HIGH, MEDIUM, and LOW:

BEGIN
  CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'TPURGENT', shares => TRUE);
  CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'TP', shares => TRUE);
  CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'HIGH', shares => TRUE);
  CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'MEDIUM', shares => TRUE);
  CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(consumer_group => 'LOW', shares => TRUE);
END;
/

See CS_RESOURCE_MANAGER Package for more information.
Manage Runaway SQL Statements on Autonomous Database

Specifies how you configure Autonomous Database to terminate SQL statements automatically based on their query runtime or their IO usage.

You can set runtime run-away rules for query run time and IO usage in Database Actions or using the PL/SQL package `CS_RESOURCE_MANAGER`.

Follow these steps to use Database Actions to set runtime usage rules:

1. Access Database Actions as the ADMIN user.
   
   See Access Database Actions as ADMIN for more information.


3. Select the Run-away criteria tab to set usage rules for a consumer group.

4. Select the Consumer group.

5. Set runaway criteria values:
   - Query run time (seconds)
   - Amount of IO (MB)

6. Click Save Changes.

7. Click OK.

When a SQL statement in the specified consumer group runs more than the specified runtime limit or does more IO than the specified amount, then the SQL statement will be terminated.

Click Load Default Values to load the default values; then click Save Changes to apply the populated values.

You can also use the procedure `CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE` to set these rules. For example, to set a runtime limit of 120 seconds and an IO limit of 1000MB for the HIGH consumer group run the following command when connected to the database as the ADMIN user:

```
BEGIN
    CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'HIGH',
      io_megabytes_limit => 1000, elapsed_time_limit => 120);
END;
/
```

To reset the values and lift the limits, you can set the values to null:

```
BEGIN
    CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(consumer_group => 'HIGH',
      io_megabytes_limit => null, elapsed_time_limit => null);
END;
/
```

See `CS_RESOURCE_MANAGER Package` for more information.
Manage Optimizer Statistics on Autonomous Database

Describes Autonomous Database commands to run when you need to gather optimizer statistics or enable optimizer hints.

There are differences, depending on your workload: Data Warehouse, Transaction Processing, or JSON Database. See the following section appropriate to your workload:

- Manage Optimizer Statistics with Data Warehouse Workloads
- Manage Optimizer Statistics with Transaction Processing and JSON Database Workloads

Manage Optimizer Statistics with Data Warehouse Workloads

Describes Autonomous Database commands to run when you need to gather optimizer statistics or enable optimizer hints with Data Warehouse workloads.

Managing Optimizer Statistics with Data Warehouse Workloads

Autonomous Database with Data Warehouse workloads gathers optimizer statistics automatically for tables loaded with direct path operations issued in SQL (direct path load operations that bypass the SQL data processing, such as SQL*Loader direct path, do not collect statistics). For example, for loads using the DBMS_CLOUD package the database gathers optimizer statistics automatically.

If you have tables modified using conventional DML operations you can run commands to gather optimizer statistics for those tables. For example, for the SH schema you can gather statistics for all tables in the schema using the following command:

```
BEGIN
  DBMS_STATS.GATHER_SCHEMA_STATS('SH', options=>'GATHER AUTO');
END;
/
```

This example gathers statistics for all tables that have stale statistics in the SH schema.

For more information about direct-path loads see Loading Tables.

For more information on optimizer statistics see Database Concepts.

Managing Optimizer Hints with Data Warehouse Workloads

Autonomous Database with Data Warehouse ignores optimizer hints and PARALLEL hints in SQL statements by default. If your application relies on hints you can enable optimizer hints by setting the parameter OPTIMIZER_IGNORE_HINTS to FALSE at the session or system level using ALTER SESSION or ALTER SYSTEM. For example, the following command enables hints in your session:

```
ALTER SESSION
  SET OPTIMIZER_IGNORE_HINTS=FALSE;
```
You can also enable PARALLEL hints in your SQL statements by setting OPTIMIZER_IGNORE_PARALLEL_HINTS to FALSE at the session or system level using ALTER SESSION or ALTER SYSTEM. For example, the following command enables PARALLEL hints in your session:

```
ALTER SESSION
    SET OPTIMIZER_IGNORE_PARALLEL_HINTS=FALSE;
```

Manage Optimizer Statistics with Transaction Processing and JSON Database Workloads

Describes Autonomous Database commands to run when you need to gather optimizer statistics or enable optimizer hints.

Managing Optimizer Statistics with Transaction Processing and JSON Database Workloads

Autonomous Database gathers optimizer statistics automatically so that you do not need to perform this task manually and this helps to ensure your statistics are current. Automatic statistics gathering is enabled in Autonomous Database and runs in a standard maintenance window.

**Note:**

The automatic statistics gathering maintenance window is different than the maintenance window on the Oracle Cloud Infrastructure console. The Oracle Cloud Infrastructure maintenance window shows system patching information.

For more information on automatic statistics gathering maintenance window times and automatic optimizer statistics collection, see *Database Administrator’s Guide*.

For more information on optimizer statistics see *SQL Tuning Guide*.

Managing Optimizer Hints with Transaction Processing and JSON Database Workloads

Autonomous Database with Transaction Processing and JSON Database workloads honors optimizer hints and PARALLEL hints in SQL statements by default. You can disable optimizer hints by setting the parameter OPTIMIZER_IGNORE_HINTS to TRUE at the session or system level using ALTER SESSION or ALTER SYSTEM. For example, the following command disables hints in your session:

```
ALTER SESSION
    SET OPTIMIZER_IGNORE_HINTS=TRUE;
```

You can also disable PARALLEL hints in your SQL statements by setting OPTIMIZER_IGNORE_PARALLEL_HINTS to TRUE at the session or system level using ALTER SESSION or ALTER SYSTEM.

```
ALTER SESSION
    SET OPTIMIZER_IGNORE_PARALLEL_HINTS=TRUE;
```
Manage Automatic Indexing on Autonomous Database

Automatic indexing automates the index management tasks in Autonomous Database. Auto Indexing is disabled by default in Autonomous Database.

Creating indexes manually requires deep knowledge of the data model, application, and data distribution. In the past, DBAs were responsible for making choices about which indexes to create, and then sometimes the DBAs did not revise their choices or maintain indexes as the conditions changed. As a result, opportunities for improvement were lost, and use of unnecessary indexes could be a performance liability. The automatic indexing feature in Autonomous Database monitors the application workload and creates and maintains indexes automatically.

To enable automatic indexing:

1. Use the `DBMS_AUTO_INDEX.CONFIGURE` procedure to enable automatic indexing:

   ```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:**

   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');
   ```

When you use SODA with Autonomous Database the following restrictions apply:

- Automatic indexing is not supported for SQL and PL/SQL code that uses the SQL/JSON function `json_exists`. See SQL/JSON Condition `JSON_EXISTS` for more information.
- Automatic indexing is not supported for SODA query-by-example (QBE).

See Managing Auto Indexes for more information.

Manage Automatic Partitioning on Autonomous Database

Automatic partitioning analyzes and automates partition creation for tables and indexes of a specified schema to improve performance and manageability in
Autonomous Database. Automatic partitioning, when applied, is transparent and does not require any user interaction or maintenance.

Topics
• About Automatic Partitioning
• How Automatic Partitioning Works
• Configure Automatic Partitioning
• Use Automatic Partitioning
• Generate Automatic Partitioning Reports
• Example Automatic Partitioning Scenarios
• Data Dictionary Views for Automatic Partitioning

Note:
Automatic partitioning does not interfere with existing partitioning strategies and is complementary to manual partitioning in Autonomous Database. Manually partitioned tables are excluded as candidates for automatic partitioning.

About Automatic Partitioning

Automatic partitioning in Autonomous Database analyzes the application workload and automatically applies partitioning to tables and their indexes to improve performance or to allow better management of large tables.

Finding appropriate partitioning strategies requires deep knowledge of the application workload and the data distribution. When you perform manual partitioning, you must analyze your workload and make choices about how to apply partitioning to tables and indexes to improve the performance of applications. Automatic partitioning enables Autonomous Database users to benefit from partitioning without performing manual schema and workload analysis.

Automatic partitioning uses a single-column partition key combined with single-level partitioning. Automatic partitioning does not support more complex partitioning strategies such as multi-column partitioned tables or composite partitioning.

Automatic partitioning chooses from the following partition methods:
• AUTOMATIC INTERVAL: This choice is best suited for ranges of partition key values.
• LIST AUTOMATIC: This partitioning method applies to distinct partition key values.
• HASH: Applies partitioning on the partition key’s hash values.

See Partitioning Concepts for more information.

Automatic partitioning provides the following functionality:
• Analyzes application workload and finds the optimal partitioning strategy to improve query performance for tables eligible for automatic partitioning.
• Provides PL/SQL APIs for configuring automatic partitioning in a database, generating reports about partitioning findings, and generating and applying an identified partitioning strategy for a given workload.
How Automatic Partitioning Works

Unlike automatic indexing, automatic partitioning does not run periodically as a background task. Automatic partitioning only runs when you invoke it using the `DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD` function.

When invoked, automatic partitioning identifies candidate tables for automatic partitioning, evaluates partition schemes, and implements a partitioning strategy.

When you invoke automatic partitioning it performs the following tasks:

1. Identifies candidate tables for automatic partitioning by analyzing the workload for selected candidate tables.
   
   By default, automatic partitioning uses the workload information collected in an Autonomous Database for analysis. Depending on the size of the workload, a sample of queries might be considered.

2. Evaluates partition schemes based on workload analysis, quantification, and verification of the performance benefits:
   
   a. Candidate empty partition schemes with synthesized statistics are created internally and analyzed for performance.
   
   b. The candidate scheme with the highest estimated IO reduction is chosen as the optimal partitioning strategy and is internally implemented to test and verify performance.
   
   c. If a candidate partition scheme does not improve performance beyond specified performance and regression criteria, automatic partitioning is not recommended.

3. Implements the optimal partitioning strategy, if configured to do so, for the tables analyzed by the automatic partitioning procedures.

Configure Automatic Partitioning

Use the `DBMS_AUTO_PARTITION.CONFIGURE` procedure to configure automatic partitioning options.

Enable and implement recommendations

```
EXEC DBMS_AUTO_PARTITION.CONFIGURE('AUTO_PARTITION_MODE','IMPLEMENT');
```

Enable recommendations, but do not implement those recommendations

```
EXEC DBMS_AUTO_PARTITION.CONFIGURE('AUTO_PARTITION_MODE','REPORT ONLY');
```
Disable new recommendations and implementation of those new recommendations

EXEC DBMS_AUTO_PARTITION.CONFIGURE('AUTO_PARTITION_MODE', 'OFF');

**Note:**
This mode does not disable existing automatically partitioned tables.

Manage schemas and tables for automatic partitioning

Use the `AUTO_PARTITION_SCHEMA` and `AUTO_PARTITION_TABLE` settings to specify schemas and tables considered for automatic partitioning.

**Note:**
When automatic partitioning is invoked, all schemas and tables in user-managed schemas are considered for automatic partitioning if both the inclusion and exclusion lists are empty.

- Assuming the inclusion list and the exclusion list are empty, add the `HR` schema and the `SH.SALES` table to the exclusion list, preventing only those objects from automatic partitioning analysis.

```sql
BEGIN
    DBMS_AUTO_PARTITION.CONFIGURE(
        PARAMETER_NAME  => 'AUTO_PARTITION_SCHEMA',
        PARAMETER_VALUE => 'HR',
        ALLOW           => FALSE);
    DBMS_AUTO_PARTITION.CONFIGURE(
        PARAMETER_NAME  => 'AUTO_PARTITION_TABLE',
        PARAMETER_VALUE => 'SH.SALES',
        ALLOW           => FALSE);
END;
/
```

- After the previous example runs, use the following to remove the `HR` schema from the exclusion list.

```sql
BEGIN
    DBMS_AUTO_PARTITION.CONFIGURE(
        PARAMETER_NAME  => 'AUTO_PARTITION_SCHEMA',
        PARAMETER_VALUE => 'HR',
        ALLOW           => NULL);
END;
/
```
• Use the following command to remove all schemas from the exclusion list.

```sql
BEGIN
    DBMS_AUTO_PARTITION.CONFIGURE(
        PARAMETER_NAME  => 'AUTO_PARTITION_SCHEMA',
        PARAMETER_VALUE => NULL,
        ALLOW           => TRUE);
END;
/```

• Assuming the inclusion and exclusion lists are empty, the following example adds the `HR` schema to the inclusion list. As soon as the inclusion list is no longer empty, only schemas in the inclusion list are considered.

With this example, only the `HR` schema is a candidate for automatic partitioning.

```sql
BEGIN
    DBMS_AUTO_PARTITION.CONFIGURE(
        PARAMETER_NAME  => 'AUTO_PARTITION_SCHEMA',
        PARAMETER_VALUE => 'HR',
        ALLOW           => TRUE);
END;
/```

**Manage Automatic Partitioning Report Retention Period**

Set the retention period for automatic partitioning reports to 365 days.

```sql
BEGIN
    DBMS_AUTO_PARTITION.CONFIGURE(
        PARAMETER_NAME  => 'AUTO_PARTITION_REPORT_RETENTION',
        PARAMETER_VALUE => '365');
END;
/
```

See `CONFIGURE Procedure` for more information.

**Use Automatic Partitioning**

Describes the flow and general processes for using and managing automatic partitioning in Autonomous Database.

1. Choose the database for automatic partitioning.
   
   In general, Oracle recommends using automatic partitioning in cloned or manually created databases rather than production databases. The analysis and verification of automatic partitioning using `RECOMMEND_PARTITION_METHOD` is potentially a resource-intensive and long running operation that can add undesirable processing to your database.

   To use a secondary database for automatic partitioning analysis, the database must have information about your workload in an internally managed SQL workload repository.

   a. Use a cloned database for automatic partitioning.
Autonomous Database automatically collects workload information over time in an internally managed SQL workload repository maintained in the SQL Tuning Set (SYS_AUTO_STS). If you clone your production database after having run the workload for a while, the clone will have the necessary workload information. You can use automatic partitioning with such clones without any additional actions.

See Clone an Autonomous Database Instance

b. Use other databases for automatic partitioning.
You can run your workload manually to collect the necessary workload information. If you manually run your workload prior to using automatic partitioning, any Autonomous Database that contains your desired schemas and data can be used for automatic partitioning after your workload was run, regardless of whether it is cloned or manually created.

2. Recommend automatic partitioning.
Use RECOMMEND_PARTITION_METHOD to analyze your database, specific schemas, or specific tables to identify the optimal partitioning strategy, if any. The recommendation analyzes your workload and schemas verifying performance benefits by running your workload against an internally created auxiliary table. This can be a resource-intensive and long running operation, requiring CPU and IO to create the auxiliary table and verify performance. You will also temporarily need additional space, of 1 - 1.5 times, your largest candidate table.

3. Apply the recommendation.
Any recommendation can be implemented with the APPLY_RECOMMENDATION procedure in the database where the recommendation analysis occurred. Alternatively, any recommendation can be extracted from the database used for analysis and applied to any database, such as a production system. The script needed for manual modification is stored in column MODIFY_TABLE_DDL in the DBA_AUTO_PARTITION_RECOMMENDATION view.

Oracle recommends applying automatic partitioning to your database at off-peak time. While your tables will be modified to automatically partitioned tables, the conversion adds additional resource requirements to your system, such as additional CPU and IO. Automatic partitioning requires as much as 1.5 times the size of the table to being modified as additional free space, depending on concurrent ongoing DML operations on those tables.

Generate Automatic Partitioning Reports

Generate automatic partitioning reports using the REPORT_ACTIVITY and REPORT_LAST_ACTIVITY functions of the DBMS_AUTO_PARTITION package.

Generate a report, in plain text format, of automatic partitioning operations for a specific period

This example generates a report containing typical information about the automatic partitioning operations for the last 24 hours. The report is generated in plain text format by default.

DECLARE
  Report clob := NULL
BEGIN
  Report := DBMS_AUTO_PARTITION.REPORT_ACTIVITY();
END;
/

---

Chapter 28
Manage Automatic Partitioning on Autonomous Database
28-29
Generate a report, in HTML format, of automatic partitioning operations for MAY 2021

This example generates a report containing basic information about the automatic partitioning operations for the month of MAY 2021. The report is generated in the HTML format, and it includes only a summary of automatic partitioning operations.

```
DECLARE
    Report clob := NULL
BEGIN
    Report := DBMS_AUTO_PARTITION.REPORT_ACTIVITY(
        ACTIVITY_START => TO_TIMESTAMP('2021-05-01', 'YYYY-MM-DD'),
        ACTIVITY_END   => TO_TIMESTAMP('2021-06-01', 'YYYY-MM-DD'),
        TYPE           => 'HTML',
        SECTION        => 'SUMMARY',
        LEVEL          => 'BASIC' );
END;
```

Generate a report, in plain text format, of the last automatic partitioning operation

This example generates a report containing typical information about the last automatic partitioning operation. The report is generated in the plain text format by default.

```
DECLARE
    Report clob := NULL
BEGIN
    Report := DBMS_AUTO_PARTITION.REPORT_LAST_ACTIVITY();
END;
```

See `REPORT_ACTIVITY Function` for more information.

See `REPORT_LAST_ACTIVITY Function` for more information.

Example Automatic Partitioning Scenarios

Example scenarios for automatic partitioning using the `DBMS_AUTO_PARTITION API` procedures and functions.

Generate a recommendation for a single table and manually apply the recommendation

1. Set `AUTO_PARTITION_MODE` parameter to `REPORT ONLY` to enable an automatic partitioning recommendation to be made and verified. The recommendation is not applied to the table.

```
BEGIN
    DBMS_AUTO_PARTITION.CONFIGURE(
```
PARAMETER_NAME => 'AUTO_PARTITION_MODE',
PARAMETER_VALUE => 'REPORT ONLY');
END;
/

2. Validate that TPCH.LINEITEM table is a candidate for automatic partitioning. This step is optional and recommended when you are selectively targeting single tables.

SELECT DBMS_AUTO_PARTITION.VALIDATE_CANDIDATE_TABLE(
    TABLE_OWNER => 'TPCH',
    TABLE_NAME  => 'LINEITEM')
FROM DUAL;

If the table is a valid candidate, when you invoke automatic partitioning for a recommendation analysis it returns as VALID. Otherwise, the violation criteria is shown. See VALIDATE_CANDIDATE_TABLE Function for a list of criteria for eligible candidate tables.

3. Invoke the DBMS_AUTO_PARTITION API to generate a recommendation for the TPCH.LINEITEM table.

-- DEFINE SQLPLUS BIND VARIABLE FOR RECOMMENDATION ID
VARIABLE RECOMMENDATION_ID VARCHAR2(32);
BEGIN
    :RECOMMENDATION_ID := DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD(
        TABLE_OWNER => 'TPCH',
        TABLE_NAME  => 'LINEITEM');
END;
/

The recommendation analysis and verification that you perform with DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD can be a resource-intensive and long running operation and might take considerable time. You should perform this step on a database that is not your primary production system. Oracle recommends giving the verification operation sufficient resources by choosing the HIGH service.

4. Check the recommendation. The view DBA_AUTO_PARTITION_RECOMMENDATIONS contains the information on the recommendation. In this example, check the recommended partition key and partition method.

SELECT PARTITION_METHOD, PARTITION_KEY
FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
WHERE RECOMMENDATION_ID = :RECOMMENDATION_ID;

Additionally, query the same view to get the performance analysis report generated for the workload after the table was partitioned according to the recommendation.

SELECT REPORT
FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
WHERE RECOMMENDATION_ID = :RECOMMENDATION_ID;

5. After manual validation of the recommendation, apply the recommendation. If you are applying the recommendation in the database where the recommendation analysis has
taken place, apply the recommendation by executing the `APPLY_RECOMMENDATION` procedure.

```
BEGIN
  DBMS_AUTO_PARTITION.APPLY_RECOMMENDATION(
    RECOMMENDATION_ID => :RECOMMENDATION_ID);
END;
/
```

If you want to apply the recommendation to a different database, such as your production environment, extract the modification DDL. Then, run the extracted modification DDL in your target database. The query to extract the modification DDL is as follows:

```
SELECT MODIFY_TABLE_DDL
FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
WHERE RECOMMENDATION_ID = :RECOMMENDATION_ID;
```

**Example output of modification DDL:**

```
BEGIN
  -- DBMS_AUTO_PARTITION RECOMMENDATION_ID
  C3F7A59E085C2F25E05333885A0A55EA
  -- FOR TABLE "TPCH"."LINEITEM"
  -- GENERATED AT 06/04/2021 20:52:29
  DBMS_AUTO_PARTITION.BEGIN_APPLY(EXPECTED_NUMBER_OF_PARTITIONS
  => 10);

  EXECUTE IMMEDIATE
  'ALTER TABLE "TPCH"."LINEITEM"
  MODIFY PARTITION BYLIST(SYS_OP_INTERVAL_HIGH_BOUND
  ("L_SHIPDATE", INTERVAL '10' MONTH, TIMESTAMP
  '1992-01-01 00:00:00'))
  AUTOMATIC /* SCORE=23533.11; */
  (PARTITION P_NULL VALUES(NULL))
  AUTO ONLINE PARALLEL';

  DBMS_AUTO_PARTITION.END_APPLY;
EXCEPTION WHEN OTHERS THEN
  DBMS_AUTO_PARTITION.END_APPLY;
  RAISE;
END;
```

6. Verify that the table was automatically partitioned, query the catalog views.

```
SELECT T.AUTO, T.PARTITIONING_TYPE, C.COLUMN_NAME
FROM DBA_PART_TABLES T, DBA_PART_KEY_COLUMNS C
WHERE T.OWNER = 'TPCH' AND T.TABLE_NAME = 'LINEITEM'
  AND T.OWNER = C.OWNER AND T.TABLE_NAME = C.NAME;
```
Use this query to identify when automatic partitioning was applied to a given table.

```sql
SELECT APPLY_TIMESTAMP_START, APPLY_TIMESTAMP_END
FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
WHERE TABLE_OWNER = 'TPCH' AND TABLE_NAME = 'LINEITEM';
```

See **CONFIGURE Procedure** for information.

See **VALIDATE_CANDIDATE_TABLE Function** for information.

See **RECOMMEND_PARTITION_METHOD Function** for information.

See **APPLY_RECOMMENDATION Procedure** for information.

**Generate a recommendation for eligible tables and manually apply the recommendation**

1. Set `AUTO_PARTITION_MODE` parameter to `REPORT ONLY` to enable an automatic partitioning recommendation to be made and verified. The recommendation is not applied to existing tables.

   ```sql
   BEGIN
   DBMS_AUTO_PARTITION.CONFIGURE(
     PARAMETER_NAME  => 'AUTO_PARTITION_MODE',
     PARAMETER_VALUE => 'REPORT ONLY');
   END;
   /
   ```

2. Invoke the `DBMS_AUTO_PARTITION` API to generate a recommendation table.

   ```sql
   -- DEFINE SQLPLUS BIND VARIABLE FOR RECOMMENDATION ID
   VARIABLE RECOMMENDATION_ID VARCHAR2(32);
   BEGIN
   :RECOMMENDATION_ID := DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD();
   END;
   /
   ```

   The recommendation analysis and verification is a resource-intensive and long running operation and might take considerable time. On secondary, non-production databases, Oracle recommends giving the verification sufficient resources by choosing service `HIGH`.

3. Query the `DBA_AUTO_PARTITION_RECOMMENDATIONS` view to see which tables were analyzed.

   ```sql
   SELECT TABLE_OWNER, TABLE_NAME, PARTITION_METHOD, PARTITION_KEY
   FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
   WHERE RECOMMENDATION_ID = :RECOMMENDATION_ID
   ORDER BY RECOMMENDATION_SEQ;
   ```

4. Use this query to drill-down in the report for a specific table that was analyzed in the run, the `TPCH.LINEITEM` table in this example.

   ```sql
   SELECT REPORT
   FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
   ```
WHERE RECOMMENDATION_ID = :RECOMMENDATION_ID
    AND TABLE_OWNER = 'TPCH'
    AND TABLE_NAME  = 'LINEITEM';

5. Apply the recommendation by executing the APPLY_RECOMMENDATION procedure.

BEGIN
    DBMS_AUTO_PARTITION.APPLY_RECOMMENDATION(
        RECOMMENDATION_ID => :RECOMMENDATION_ID);
END;
/

Alternately, apply recommendations for a specific table that was analyzed, the TPCH.LINEITEM table in this example.

BEGIN
    DBMS_AUTO_PARTITION.APPLY_RECOMMENDATION(
        RECOMMENDATION_ID => :RECOMMENDATION_ID,
        TABLE_OWNER       => 'TPCH',
        TABLE_NAME        => 'LINEITEM');
END;
/

See CONFIGURE Procedure for information.

See RECOMMEND_PARTITION_METHOD Function for information.

See APPLY_RECOMMENDATION Procedure for information.

---

**Note:**

Recommendations of automatic partitioning generated by the RECOMMEND_PARTITION_METHOD function have a time limit, specified by the TIME_LIMIT parameter, with a default of 1 day. If you are analyzing a large system with many candidate tables, a single invocation may not generate a recommendation for all tables that can be partitioned. You can safely invoke the recommendation for auto partitioning repeatedly to generate recommendations for additional tables. When the function is invoked and zero rows are in DBA_AUTO_PARTITION_RECOMMENDATIONS for the RECOMMENDATION_ID, then the function did not find any additional candidate tables for automatic partitioning.

---

Generate and automatically apply recommendations for eligible tables

1. Set AUTO_PARTITION_MODE parameter to REPORT ONLY to enable an automatic partitioning recommendation to be made and verified. The recommendation is not applied to existing tables.

BEGIN
    DBMS_AUTO_PARTITION.CONFIGURE(
        PARAMETER_NAME  => 'AUTO_PARTITION_MODE',
        PARAMETER_VALUE => 'IMPLEMENT');

```
2. Invoke the `DBMS_AUTO_PARTITION` API to generate a recommendation table.

   ```sql
   -- DEFINE SQLPLUS BIND VARIABLE FOR RECOMMENDATION ID
   VARIABLE RECOMMENDATION_ID VARCHAR2(32);
   BEGIN
     :RECOMMENDATION_ID := DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD();
   END;
   /
   ``

   The recommendation analysis and verification is a resource-intensive and long running operation and might take considerable time. On secondary, non-production databases, Oracle recommends giving the verification sufficient resources by choosing service `HIGH`.

3. Query the `DBA_AUTO_PARTITION_RECOMMENDATIONS` view to see which tables were analyzed.

   ```sql
   SELECT TABLE_OWNER, TABLE_NAME, PARTITION_METHOD, PARTITION_KEY
   FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
   WHERE RECOMMENDATION_ID = :RECOMMENDATION_ID
   ORDER BY RECOMMENDATION_SEQ;
   ```

4. Use the `REPORT_LAST_ACTIVITY` function to retrieve the report on the actions taken during the last run.

   ```sql
   SELECT DBMS_AUTO_PARTITION.REPORT_LAST_ACTIVITY() FROM DUAL;
   ```

See `CONFIGURE Procedure` for information.

See `RECOMMEND_PARTITION_METHOD Function` for information.

See `REPORT_LAST_ACTIVITY Function` for information.

Data Dictionary Views for Automatic Partitioning

There are two new views and one updated view in the data dictionary for information about the automatic partitioning configuration and recommendations in your database.

**DBMS_AUTO_PARTITION DBA_AUTO_PARTITION_CONFIG View**

Displays the current configuration parameter settings for automatic partitioning.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARAMETER_NAME</td>
<td>Name of the configuration parameter</td>
</tr>
<tr>
<td>PARAMETER_VALUE</td>
<td>Value of the configuration parameter</td>
</tr>
<tr>
<td>LAST_MODIFIED</td>
<td>Time, in UTC, at which the parameter value was last modified.</td>
</tr>
<tr>
<td>MODIFIED_BY</td>
<td>User who last modified the parameter value</td>
</tr>
</tbody>
</table>
DBMS_AUTO_PARTITION DBA_AUTO_PARTITION_RECOMMENDATIONS View

When you run `CONFIGURE` or `RECOMMEND_PARTITION`, the results from those procedures is stored in this view. The `RECOMMENDATION_ID` is used in several procedures and functions.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE_OWNER</td>
<td>Owner of the table</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>Owner of the table</td>
</tr>
<tr>
<td>PARTITION_METHOD</td>
<td>Recommended partition method. See <code>CONFIGURE</code> Procedure</td>
</tr>
<tr>
<td>PARTITION_KEY</td>
<td>Recommended partition key. NULL means that analysis complete and recommendation is not to partition the table.</td>
</tr>
<tr>
<td>GENERATE_TIMESTAMP</td>
<td>Time, in UTC, when this recommendation was generated.</td>
</tr>
<tr>
<td>RECOMMENDATION_ID</td>
<td>ID used with <code>DBMS_AUTO_PARTITION</code> APIs to get additional information about this recommendation.</td>
</tr>
<tr>
<td>RECOMMENDATION_SEQ</td>
<td>When a recommendation ID has recommendations for multiple tables, provides the order in which the recommendations were generated. Performance reports are generated assuming that earlier recommendations have been applied. For example, the report for <code>RECOMMENDATION_SEQ = 2</code> assumes recommendations have been applied for both <code>RECOMMENDATION_SEQ = 1</code> and <code>RECOMMENDATION_SEQ = 2</code>.</td>
</tr>
<tr>
<td>MODIFY_TABLE_DDL</td>
<td>DDL that would be, or was, used to apply the recommendation.</td>
</tr>
<tr>
<td>APPLY_TIMESTAMP_START</td>
<td>Time, in UTC, when application of this recommendation was started. NULL if recommendation was not applied.</td>
</tr>
<tr>
<td>APPLY_TIMESTAMP_END</td>
<td>Time, in UTC, when application of this recommendation was finished. NULL if recommendation was not applied or if application has not finished.</td>
</tr>
<tr>
<td>REPORT</td>
<td>SQL Performance Analyzer report from SQL execution on database after recommendation is applied.</td>
</tr>
</tbody>
</table>

DBMS_AUTO_PARTITION Updates to Existing Views

Discusses the changes to existing views as a result of the implementation of automatic partitioning.

In `*_PART_TABLES`, the `AUTO` column (`VARCHAR2(3)`) was added. Its values are as follows:

- **YES** - If the table was partitioned by `DBMS_AUTO_PARTITION`.
- **NO** - If the table was not partitioned by `DBMS_AUTO_PARTITION`.

Monitor Autonomous Database with Performance Hub

You can view real-time and historical performance data from the Performance Hub. The Performance Hub shows Active Session History (ASH) analytics, SQL monitoring and workload information.

Perform the following prerequisite steps as necessary:
• Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
• From the Oracle Cloud Infrastructure left navigation menu click Oracle Database, and then click Autonomous Database.
• From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
1. On the Autonomous Databases page select an Autonomous Database from the links under the Display Name column.
2. From the Autonomous Database Details page click Performance Hub. The Performance Hub is displayed. This page has the following sections:
   • Quick Select: Allows you to quickly select a time range.
   • Time Selector: Allows you to select a custom time range.
   • AWR Report: Reports drop-down list provides the option to create and download an Automatic Workload Repository (AWR) report.
   • The tabbed data area shows the following tabs:
     – ASH Analytics
     – SQL Monitoring
     – ADDM
     – Workload
     – Blocking Sessions

**Time Selector**

The time selector is a set of controls at the top of the Performance Hub page. You use these controls to set the time range for Performance Hub to monitor.

You can use the Quick Select list to quickly set the time range to Last Hour, Last 8 Hours, Last 24 Hours or Last Week, or you can click the Time Range field and specify a custom time range. Additionally, you can use the Time Zone list to have all times based on UTC (Coordinated Universal Time) time, your local web browser time, or the time zone setting of the database.

The Activity timeline shows active sessions during the selected time range. It displays the average number of active sessions broken down by CPU, User I/O, and Wait. It also shows the Max CPU usage.

The sliding box on the timeline is the time slider. Use the time slider to select a section of the time range.

You can slide the box to the left or the right to shift the time selection, and you can widen or narrow the box to increase or decrease the section's timespan. To slide the entire box, left-click anywhere inside the box and drag the box to the left or the right. 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 width of the time slider.

Click Refresh to refresh the data in Performance Hub according to the time range chosen.

(Required) Enter introductory text here, including the definition and purpose of the concept.
AWR Report

The Automatic Workload Repository (AWR) collects, processes, and maintains performance statistics for problem detection and self-tuning purposes. This data is both in memory and stored in the database.

An AWR report shows data captured between two points in time (or snapshots). AWR reports are divided into multiple sections. The content of the report contains the workload profile of the system for the selected range of snapshots. The HTML report includes links that you can use to navigate quickly between sections.

The statistics collected and processed by AWR include:

- Object statistics that determine both access and usage statistics of database segments
- Time model statistics based on time usage for activities, displayed in the $SYS\_TIME\_MODEL$ and $SESS\_TIME\_MODEL$ views
- Some of the system and session statistics collected in the $SYSSTAT$ and $SESSTAT$ views
- SQL statements that are producing the highest load on the system, based on criteria such as elapsed time and CPU time
- ASH statistics, representing the history of recent sessions activity

Follow these steps to generate and download an AWR report from the Performance Hub:

1. Use the time selector controls to specify a time range that includes the period for which you want to generate the AWR report.
2. Click Reports and then choose Automatic Workload Repository to display the Generate Automatic Workload Repository Report dialog.
3. Choose the Start Snapshot and End Snapshot for the report and then click Download.
4. Performance Hub generates the report, displays the name of the report file, and, depending on your browser settings, automatically downloads it to your default download location or prompts you to specify a download location.

For more information about using an AWR report, see Using the AWR Compare Periods Reports.

ASH Analytics

The ASH Analytics tab 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.

You can use the ASH Sample Resolution drop-down list to control the data resolution of the chart, from the coarsest setting of Low, which shows the fewest data points and has the largest interval between data points, to the finest setting of Maximum, which
shows the most data points and has the smallest interval between data points. Note that the higher the setting you choose, the longer it takes Performance Hub to retrieve data and display the chart.

See Active Session History (ASH) in Oracle Database Concepts for more information on Active Session History.

**SQL Monitoring**

SQL statements are only monitored if they've been running for at least five seconds or if they're running in parallel. The SQL Monitoring tab displays monitored SQL statement executions by dimensions including Last Active Time, CPU Time, and Database Time.

The SQL Monitoring 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.

**ADDM**

The Automatic Database Diagnostic Monitor tab provides access to analysis information gathered by the Automatic Database Diagnostic Monitor (ADDM) tool. ADDM analyzes AWR (Automatic Workload Repository) snapshots on a regular basis, locates root causes of any performance problems, provides recommendations for correcting the problems, and identifies non-problem areas of the system.

Because AWR is a repository of historical performance data, ADDM can analyze performance issues after the event, often saving time and resources in reproducing a problem.

After selecting the **ADDM** tab, use the time selector controls to specify a time range that includes the period for which you want to view ADDM info. Then, select which ADDM analysis dataset you want to view from the **Task Name** list.

Performance Hubs retrieves the ADDM analysis information, presenting it in two tables:

- **Findings**. This table lists performance symptoms and problems. For more information about a finding, you can click its name to display details and recommendations regarding it.
- **Warnings and Information**. This table lists findings that can help you understand the analysis:
Warning messages identify issues such as missing data in an AWR that may affect the completeness or accuracy of the ADDM analysis.

Information messages are not performance issues, but help you understand how the database is performing. This may include identification of non-problem areas of the database and automatic database maintenance activity.

For more information about ADDM findings and how to interpret them, see ADDM Analysis Results in *Oracle Database Performance Tuning Guide*.

## Workload

The workload tab shows four chart areas that show the workload on the database.

- **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.

## Blocking Sessions

The blocking sessions tab lists sessions that are waiting or are blocked by sessions that are waiting. You can set the minimum wait time required for sessions to be displayed in the list, and you can view a variety of information about a session to determine whether to let it continue or to stop it.
You access this information by clicking the links in columns of the table row for the session:

• **Lock** column: click the name of the lock type to display the Wait Event Details dialog box.

• **Wait Event** column: click the name of the wait event to display the Session Lock Information dialog box.

• **User Session** column: click the session identifier to display the Session Details page.

• **SQL ID** column: click the SQL ID to display the SQL Details page.

After researching a session, you can stop it by selecting the checkbox at the start of the session's table row and then clicking **Kill Session**. A confirmation dialog is displayed, requesting you to confirm the operation.

### Monitor the Performance of Autonomous Database with Oracle Management Cloud

Oracle Management Cloud allows you to monitor availability and performance for Autonomous Databases. You can use Oracle Database Management, part of Oracle Management Cloud, to monitor Autonomous Databases and On-premise Oracle Databases.

For information on using Oracle Management Cloud with Autonomous Database see the following:

• **Using Oracle Database Management for Autonomous Databases**

• **Getting Started with Oracle Management Cloud**

### Use Database Management Service to Monitor Databases

**APPLIES TO:** Oracle Cloud only

You can use Database Management Service to monitor the health of a single Autonomous Database or a fleet of Autonomous Databases.

Database Management Service lets you:

• Monitor the key performance and configuration metrics of a fleet of Autonomous Databases.

• Compare and analyze database metrics over a selected period.

• Group your critical Autonomous Databases, which reside across compartments, into a Database Group, and monitor them.

To enable Database Management Service:

• Perform the prerequisite steps as described in [General Prerequisite Tasks](#).

• Obtain the required permissions as described in [Permissions Required to Enable Database Management for Autonomous Databases](#).

• Obtain additional permissions to use the Database Management **Fleet Summary** and [Managed Database Details](#) pages and view alarms on the **Fleet Summary** page. These are listed in [Additional Permissions Required to Use Database Management](#).

• Finally, enable Database Management for Autonomous Databases by following the steps outlined in [Enable Database Management for Autonomous Databases](#).

After enabling Database Management Service, you can perform the actions listed below.
Monitor Performance with Autonomous Database Metrics

You can monitor the health, capacity, and performance of your databases with metrics, alarms, and notifications. You can use Oracle Cloud Infrastructure Console or Monitoring APIs to view metrics.

Topics

- View Metrics for an Autonomous Database Instance
- View Metrics for Autonomous Databases in a Compartment
- Autonomous Database Metrics and Dimensions

View Metrics for an Autonomous Database Instance

Shows the steps to view the Autonomous Database metrics.

Note:

To view metrics you must have the required access as specified in an Oracle Cloud Infrastructure policy (whether you're using the Console, the REST API, or another tool). See Getting Started with Policies for information on policies.

Perform the following steps as necessary:
• Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.

• From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.

• On the Autonomous Databases page select an Autonomous Database from the links under the Display Name column.

To view metrics for an Autonomous Database instance:

1. On the Details page, under Resources, click Metrics.

2. There is a chart for each metric. In each chart you can select the Interval and Statistic or use the default values.

The following list shows the default Oracle Cloud Infrastructure Console metrics. See Database Metrics for a list of the database metrics and dimensions.

<table>
<thead>
<tr>
<th>Metric Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Utilization</td>
<td>CPU utilization expressed as a percentage, aggregated across all consumer groups. The utilization percentage is reported with respect to the number of CPUs the database is allowed to use, which is two times the number of OCPUs.</td>
</tr>
<tr>
<td>Storage Utilization</td>
<td>The percentage of provisioned storage capacity currently in use. Represents the total allocated space for all tablespaces.</td>
</tr>
<tr>
<td>Sessions</td>
<td>The number of sessions in the database.</td>
</tr>
<tr>
<td>Execute Count</td>
<td>The number of user and recursive calls that ran SQL statements during the selected interval.</td>
</tr>
<tr>
<td>Running Statements</td>
<td>The number of running SQL statements, aggregated across all consumer groups, during the selected interval.</td>
</tr>
<tr>
<td>Queued Statements</td>
<td>The number of queued SQL statements, aggregated across all consumer groups, during the selected interval.</td>
</tr>
</tbody>
</table>
| Database Availability| The database is available for connections during the selected time interval (data for this metric lags by 1 hour). Possible values for this metric: 
  • 1 = Database is Available
  • 0 = Database is Unavailable
You can set an alarm that is triggered if the database is not available (value 0). |

Note:
Availability is calculated based on the "Monthly Uptime Percentage" described in the Oracle PaaS and IaaS Public Cloud Services Pillar Document, under Autonomous Database Availability Service Level Agreement.
### Metric Name | Description
--- | ---
Regional Availability | Shows the average availability percentage of Autonomous Databases across a set of services in each region, ensuring that regions are performing as per Service Level Agreement. Availability data is updated daily for each region.
Failed Connections | Shows the total number of failed connections to the database during the selected interval.

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.

- Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
- From the left navigation list click **Observability & Management**. Under **Monitoring**, click **Service Metrics**.

To use the metrics service to view Autonomous Database metrics:

1. On the Service Metrics page, under **Compartment** select your compartment.
2. On the Service Metrics page, under **Metric Namespace** select **oci_autonomous_database**.
3. If there are multiple Autonomous Databases in the compartment you can show metrics aggregated across the Autonomous Databases by selecting **Aggregate Metric Streams**.
4. If you want to limit the metrics you see, next to **Dimensions** click **Add** (click **Edit** if you have already added dimensions).
   a. In the **Dimension Name** field select a dimension.
   b. In the **Dimension Value** field select a value.
   c. Click **Done**.
   In the Edit dimensions dialog click **+Additional Dimension** to add an additional dimension. Click x to remove a dimension.

To create an alarm on a specific metric, click **Options** and select **Create an Alarm on this Query**. See **Managing Alarms** for information on setting and using alarms.
Autonomous Database Metrics and Dimensions

You can limit the instances where you see metrics with dimensions. The available dimensions include: workload type, instance display name, region, and the instance OCID.

Use dimensions by selecting values in the Oracle Cloud Infrastructure Console Service Metrics page or by setting dimension values with the API. See View Metrics for Autonomous Databases in a Compartment to view metrics and to select metric dimensions.

See Database Metrics for a list of the database metrics and dimensions.

Perform SQL Tracing on Autonomous Database

Use SQL tracing to help you identify the source of an excessive database workload, such as a high load SQL statement in your application.

Topics

• Configure SQL Tracing on Autonomous Database
• Enable SQL Tracing on Autonomous Database
• Disable SQL Tracing on Autonomous Database
• View Trace File Saved to Cloud Object Store on Autonomous Database
• View Trace Data in SESSION_CLOUD_TRACE View on Autonomous Database

Configure SQL Tracing on Autonomous Database

Shows the steps to configure SQL tracing on Autonomous Database.

Note:

If you enable SQL Tracing your application performance for the session may be degraded while the trace collection is enabled. This negative performance impact is expected due to the overhead of collecting and saving trace data.

To configure your database for SQL tracing, do the following:

1. Create a bucket to store trace files in your Cloud Object Storage.

   To save the SQL tracing files, the bucket can be in any Cloud Object Store that Autonomous Database supports.

   For example, to create a bucket in Oracle Cloud Infrastructure Object Storage, do the following

   a. Open the Oracle Cloud Infrastructure Console.
   b. Select Storage from the menu.
   c. Under Storage, select Object Storage and Archive Storage.
   d. Click Create Bucket.
   e. In the Create Bucket page, enter the Bucket Name and click Create.
If you are using an Oracle Cloud Infrastructure Object Storage, note that SQL tracing files are only supported with buckets created in the standard storage tier, make sure you pick **Standard** as the storage tier when creating your bucket. For information on the Standard Object Storage Tier, see *Overview of Object Storage*.

2. Create a credential for your Cloud Object Storage account using `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

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

   See *CREATE_CREDENTIAL Procedure* for details on the arguments for `username` and `password` parameters for different object storage services.

3. Set initialization parameters to specify the Cloud Object Storage URL for a bucket for SQL trace files and to specify the credentials to access the Cloud Object Storage.

   a. Set database property `DEFAULT_LOGGING_BUCKET` to specify the logging bucket on Cloud Object Storage.

      For example, if you create the bucket with Oracle Cloud Infrastructure Object Storage:

      ```sql
      SET DEFINE OFF;
      ALTER DATABASE PROPERTY SET
          DEFAULT_LOGGING_BUCKET = 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/bucket_name/o/';
      ```

      *Where namespace-string* is the Oracle Cloud Infrastructure Object Storage namespace and *bucket_name* is the name of the bucket you previously created. See *Understanding Object Storage Namespaces* for more information.

      See *Regions and Availability Domains* for a list of regions.

      The Cloud Object Store you use for SQL Tracing files can be any Cloud Object Store that Autonomous Database supports.

   b. Set the database property `DEFAULT_CREDENTIAL` to the credential you created in Step 2.

      For example:

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

      Including the schema name with the credential is required. In this example the schema is "ADMIN".
Enable SQL Tracing on Autonomous Database

Shows the steps to enable SQL tracing for the database session.

**Note:**

If you enable SQL tracing your application performance for the session may be degraded while the trace collection is enabled. This negative performance impact is expected due to the overhead of collecting and saving trace data.

Before you enable SQL tracing you must configure the database to save SQL Trace files. See [Configure SQL Tracing on Autonomous Database](#) for more information.

To enable SQL tracing, do the following:

1. *(Optional)* Set a client identifier for the application. This step is optional but is recommended. SQL tracing uses the client identifier as a component of the trace file name when the trace file is written to Cloud Object Store.
   
   For example:
   
   ```sql
   BEGIN
   DBMS_SESSION.SET_IDENTIFIER('sqlt_test');
   END;
   /```

2. *(Optional)* Set a module name for the application. This step is optional but is recommended. SQL tracing uses the module name as a component of the trace file name when the trace file is written to Cloud Object Store.
   
   For example:
   
   ```sql
   BEGIN
   DBMS_APPLICATION_INFO.SET_MODULE('modname', null);
   END;
   /```

3. Enable the SQL Trace facility.

   ```sql
   ALTER SESSION SET SQL_TRACE = TRUE;
   ```

4. Run your workload.

   This step involves running the entire application or specific parts of the application. While you run your workload in the database session, SQL tracing data is collected.

5. Disable SQL Tracing.

   When you disable SQL tracing the collected data for the session is written to a table in your session and to a trace file in the bucket you configure when you set up SQL tracing. See [Disable SQL Tracing on Autonomous Database](#) for details.
Disable SQL Tracing on Autonomous Database

Shows the steps to disable SQL tracing on Autonomous Database.

To disable SQL tracing, do the following:

1. Disable the SQL Trace facility.

   ALTER SESSION SET SQL_TRACE = FALSE;

2. (Optional) as needed for your environment, you may want to reset the database property `DEFAULT_LOGGING_BUCKET` to clear the value for the logging bucket on Cloud Object Storage.

   For example:

   ALTER DATABASE PROPERTY SET DEFAULT_LOGGING_BUCKET = '';

When you disable SQL tracing, the tracing data collected while the session runs with tracing enabled is copied to a table and sent to a trace file on Cloud Object Store. You have two options to view trace data:

- View and analyze SQL Trace data in the trace file saved to Cloud Object Store. See View Trace File Saved to Cloud Object Store on Autonomous Database for more information.

- View and analyze SQL Trace data saved to the view `SESSION_CLOUD_TRACE`. See View Trace Data in SESSION_CLOUD_TRACE View on Autonomous Database for more information.

View Trace File Saved to Cloud Object Store on Autonomous Database

Describes the output file naming for SQL trace files and shows the commands to use TKPROF to organize and view trace file data.

You use SQL trace file data to analyze application performance on Autonomous Database. When you disable SQL trace in your database session, data is written to the Cloud Object Store bucket configured with `DEFAULT_LOGGING_BUCKET`.

The SQL Trace facility writes the trace data collected in the session to Cloud Object Store in the following format:

`default_logging_bucket/sqltrace/clientID/moduleName/sqltrace_numID1_numID2.trc`

The components of the file name are:

- `default_logging_bucket`: is the value of the `DEFAULT_LOGGING_BUCKET` database property. See Configure SQL Tracing on Autonomous Database for more information.

- `clientID`: is the client identifier. See Enable SQL Tracing on Autonomous Database for more information.
moduleName is the module name. See Enable SQL Tracing on Autonomous Database for more information.

numID1_numID2 are two identifiers that the SQL Trace facility provides. The numID1 and numID2 numeric values uniquely distinguish each trace file name from other sessions using tracing and creating trace files in the same bucket in the Cloud Object Storage.

When the database service supports parallelism and a session runs a parallel query, the SQL Trace facility can produce multiple trace files with different numID1 and numID2 values.

Note:

When SQL tracing is enabled and disabled multiple times within the same session, each trace iteration generates a separate trace file in Cloud Object Store. To avoid overwriting previous traces that were generated in the session, subsequently generated files follow the same naming convention and add a numeric suffix to the trace file name. This numeric suffix starts with the number 1 and is incremented by 1 for each tracing iteration thereafter.

For example, the following is a sample generated trace file name when you set the client identifier to "sql_test" and the module name to "modname":

sqltrace/sqlt_test/modname/sqltrace_5415_56432.trc

You can run TKPROF to translate the trace file into a readable output file.

1. Copy the trace file from Object Store to your local system.
2. Navigate to the directory in which the trace file is saved.
3. Run the TKPROF utility from the operating system prompt using the following syntax:

   tkprof filename1 filename2 [waits=yes|no] [sort=option] [print=n]
   [aggregate=yes|no] [insert=filename3] [sys=yes|no] [table=schema.table]
   [explain=user/password] [record=filename4] [width=n]

   The input and output files are the only required arguments.
4. To view online Help, invoke TKPROF without arguments.

See "Tools for End-to-End Application Tracing" in Oracle Database SQL Tuning Guide for information about using the TKPROF utility.
View Trace Data in SESSION_CLOUD_TRACE View on Autonomous Database

When you enable SQL Tracing, the same trace information that is saved to the trace file on Cloud Object Store is available in the SESSION_CLOUD_TRACE view in the session where the tracing was enabled.

While you are still in the database session you can view SQL tracing data in the SESSION_CLOUD_TRACE view. The SESSION_CLOUD_TRACE view includes two columns: ROW_NUMBER and TRACE:

DESC SESSION_CLOUD_TRACE

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROW_NUMBER</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>TRACE</td>
<td></td>
<td>VARCHAR2(32767)</td>
</tr>
</tbody>
</table>

The ROW_NUMBER specifies the ordering for trace data found in the TRACE column. Each line of trace output written to a trace file becomes a row in the table and is available in the TRACE column.

After you disable SQL tracing for the session, you can run queries on the SESSION_CLOUD_TRACE view.

For example:

SELECT trace FROM SESSION_CLOUD_TRACE ORDER BY row_number;

The data in SESSION_CLOUD_TRACE persists for the duration of the session. After you log out or close the session, the data is no longer available.

If SQL Trace is enabled and disabled multiple times within the same session, SESSION_CLOUD_TRACE shows the trace data for all the iterations cumulatively. Thus, re-enabling tracing in a session after previously disabling tracing does not remove the trace data produced by the earlier iteration.

Monitor Autonomous Database Availability

Shows you the steps to view availability information for an Autonomous Database instance.

Database Availability is calculated based on the "Monthly Uptime Percentage" described in the Oracle PaaS and IaaS Public Cloud Services Pillar Document, under Autonomous Database Availability Service Level Agreement.

Perform the following prerequisite steps as necessary:

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

• On the Autonomous Databases page select an Autonomous Database from the links under the **Display Name** column.

To view Database Availability for an Autonomous Database instance:

1. On the **Details** page under General Information, click the Database Availability link in the **Lifecycle State** field.

   ![Database Availability](image)

   - The service level agreement (SLA) metrics are for informational purposes only. This database’s availability SLA is at least 99.95%. [Learn more](https://example.com) about Oracle Cloud SLAs.
   - **Month & Year** | **Downtime** | **Availability Percentage**
   - August 2022 | 0 minutes | 100.0000 %
   - July 2022 | 0 minutes | 100.0000 %
   - June 2022 | 0 minutes | 100.0000 %

   Click **Close** to dismiss the page.

   Note the following for the Autonomous Database instance availability:

   • For databases with Cross-Region Autonomous Data Guard enabled, review both the primary database and the remote standby database availability.
   • The Database Availability page shows information for the current month and for the previous two months.
   • After you provision a new Autonomous Database instance, database availability information is only available for the current month.
   • See **View Metrics for an Autonomous Database Instance** for more information on database availability. The Database Availability metric provides the Database Availability history for an Autonomous Database instance.
   • Database availability information is not available prior to the introduction of this feature in April 2022.

   See [Monitor Regional Availability of Autonomous Databases](https://example.com) for information on regional average availability metrics for Autonomous Database.
Monitor Regional Availability of Autonomous Databases

The regional availability is represented as a list of daily uptime percentages of Autonomous Databases across a set of services in each available region.

The regional availability data is updated daily on the following page:

Regional Average Availability Metrics for Oracle Autonomous Database

Notes for regional availability:

- The regional availability page shows information for the current month and for the previous two months.
- Regional availability information starts in June 2022 (availability data is not shown for dates prior to June 2022).

See View Metrics for an Autonomous Database Instance for more information regional availability.

See Monitor Autonomous Database Availability for information on Autonomous Database instance availability.

Service Console Replacement with Database Actions

The Autonomous Database functionality that was available from the Service Console is now available in Database Actions. If you are familiar with the Service Console, the following provides a mapping of equivalent functionality in Database Actions.

<table>
<thead>
<tr>
<th>Service Console Area</th>
<th>Equivalent in Database Actions</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>Monitoring: Database Monitor → Overview tab</td>
<td>Database Monitor Overview</td>
</tr>
<tr>
<td>Activity: Monitor Tab</td>
<td>Monitoring: Database Monitor → Monitor tab</td>
<td>Database Monitor Activity</td>
</tr>
<tr>
<td>Activity: Monitored SQL Tab</td>
<td>Monitoring: Performance Hub</td>
<td>Use Database Actions to Monitor Active Session History Analytics and SQL Statements</td>
</tr>
<tr>
<td>Administration: Download Client Credentials (Wallet)</td>
<td>Administration: Download Client Credentials (Wallet)</td>
<td>Download Client Credentials (Wallets)</td>
</tr>
<tr>
<td>Administration: Set Administrator Password</td>
<td>Administration: Database Users</td>
<td>Manage the Administrator Account on Autonomous Database</td>
</tr>
<tr>
<td>Administration: Manage Oracle ML Users</td>
<td>Development: Oracle Machine Learning</td>
<td>Create and Update User Accounts for Oracle Machine Learning Components on Autonomous Database</td>
</tr>
</tbody>
</table>
### Available Metrics: oci_autonomous_database

This topic describes the metrics emitted by the Database service in the `oci_autonomous_database` namespace.

Database service metrics for Autonomous Databases include the following **dimensions**:

- **AUTONOMOUSDBTYPE**
  The type of Autonomous Database, Autonomous Data Warehouse (ADW) or Autonomous Transaction Processing (ATP).

- **deploymentType**
  The Exadata infrastructure type, shared or dedicated. When using the Console to view default metric charts for multiple Autonomous Databases, you must specify this dimension.

This topic covers the shared Exadata infrastructure metrics. See Monitor Databases with Autonomous Database Metrics for information on dedicated Exadata infrastructure metrics.

- **DISPLAYNAME**
  The friendly name of the Autonomous Database.

- **REGION**
  The region in which the Autonomous Database resides.

- **RESOURCEID**
  The OCID of the Autonomous Database.

- **RESOURCENAME**
  The name of the Autonomous Database.

### Service Console Area | Equivalent in Database | More Information
---|---|---
Development: Download Oracle Instant Client | Downloads: Download Oracle Instant Client | Connect to Autonomous Database Using a Client Application
Development: Download SODA Drivers | Downloads: Download SODA Drivers | Use SODA for REST with Autonomous Database
Development: Oracle APEX | Development: APEX | Access Oracle APEX Administration Services
Development: Database Actions | Database Actions | Connect with Built-in Oracle Database Actions
Development: RESTful Services and SODA | Related Services: RESTful Services and SODA | Access RESTful Services and SODA for REST
Development: Oracle Machine Learning RESTful services | Related Services: Oracle Machine Learning RESTful services | REST API for Oracle Machine Learning Services
Development: Oracle Database API for MongoDB | Related Services: Oracle Database API for MongoDB | Connect MongoDB Applications to Autonomous Database
The metrics listed in the following table are automatically available for any Autonomous Database that you create. You do not need to enable monitoring on the resource to get these metrics.

Note:
Valid alarm intervals are 5 minutes or greater due to the frequency at which these metrics are emitted. See To create an alarm for details on creating alarms.

In the following table, metrics that are marked with an asterisk (*) can be viewed only on the Service Metrics page of the Oracle Cloud Infrastructure console. All metrics can be filtered by the dimensions described in this topic.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Metric Display Name</th>
<th>Unit</th>
<th>Description</th>
<th>Collection Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionLatency</td>
<td>Connection Latency</td>
<td>Milliseconds</td>
<td>The time taken to connect to a Autonomous Database that uses shared Exadata infrastructure in each region from a Compute service virtual machine in the same region. Statistic: Max Interval: 5 minutes</td>
<td>5 minutes</td>
</tr>
<tr>
<td>CpuUtilization</td>
<td>CPU Utilization</td>
<td>percent</td>
<td>The CPU usage expressed as a percentage, aggregated across all consumer groups. The utilization percentage is reported with respect to the number of CPUs the database is allowed to use, which is two times the number of OCPUs. Statistic: Mean Interval: 1 minute</td>
<td>NA</td>
</tr>
<tr>
<td>Metric</td>
<td>Metric Display Name</td>
<td>Unit</td>
<td>Description</td>
<td>Collection Frequency</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>CurrentLogons*</td>
<td>Current Logons</td>
<td>count</td>
<td>The number of successful logons during the selected interval.</td>
<td>1 minute</td>
</tr>
</tbody>
</table>
| DatabaseAvailability | Database Availability       | count | The database is available for connections in the given minute, with possible values:  
|                      |                             |       | • 1 = DB Available  
|                      |                             |       | • 0 = DB Unavailable  
|                      |                             |       | Data for this metric lags by 1 hour.                                        |                      |
| DBBlockChanges       | DB Block Changes            | count | The number of changes that were part of an update or delete operation that were made to all blocks in the SGA. Such changes generate redo log entries and thus become permanent changes to the database if the transaction is committed. This approximates total database work. This statistic indicates the rate at which buffers are being dirtied, during the selected time interval.  
<p>|                      |                             |       | 1 minute                                                                   |                      |</p>
<table>
<thead>
<tr>
<th>Metric</th>
<th>Metric Display Name</th>
<th>Unit</th>
<th>Description</th>
<th>Collection Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBTime*</td>
<td>DB Time</td>
<td>seconds per second</td>
<td>The amount of time database user sessions spend executing database code (CPU Time + WaitTime). DB Time is used to infer database call latency, because DB Time increases in direct proportion to both database call latency (response time) and call volume.  It is calculated as the average rate of accumulation of database time by foreground sessions in the database over the time interval. Also known as Average Active Sessions.  Statistic: Mean Interval: 1 minute</td>
<td>NA</td>
</tr>
<tr>
<td>ExecuteCount</td>
<td>Execute Count</td>
<td>count</td>
<td>The number of user and recursive calls that executed SQL statements during the selected interval. Statistic: Sum Interval: 1 minute</td>
<td>NA</td>
</tr>
<tr>
<td>FailedConnections*</td>
<td>Failed Connections</td>
<td>count</td>
<td>The number of failed database connections. Statistic: Sum Interval: 1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>Metric</td>
<td>Metric Display Name</td>
<td>Unit</td>
<td>Description</td>
<td>Collection Frequency</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>FailedLogons</td>
<td>Failed Logons</td>
<td>count</td>
<td>The number of log ons that failed because of an invalid user name and/or password, during the selected interval. Statistic: Mean</td>
<td>1 minute</td>
</tr>
<tr>
<td>HardParseCount</td>
<td>Parse Count (Hard)</td>
<td>count</td>
<td>The number of parse calls (real parses) during the selected time interval. A hard parse is an expensive operation in terms of memory use, because it requires Oracle to allocate a workheap and other memory structures and then build a parse tree. Statistic: Sum</td>
<td>1 minute</td>
</tr>
<tr>
<td>LogicalReads</td>
<td>Session Logical Reads</td>
<td>count</td>
<td>The sum of &quot;db block gets&quot; plus &quot;consistent gets&quot;, during the selected time interval. This includes logical reads of database blocks from either the buffer cache or process private memory.</td>
<td>1 minute</td>
</tr>
<tr>
<td>ParseCount*</td>
<td>Parse Count (Total)</td>
<td>count</td>
<td>The number of hard and soft parses during the selected interval. Statistic: Sum</td>
<td>1 minute</td>
</tr>
<tr>
<td>Metric</td>
<td>Metric Display Name</td>
<td>Unit</td>
<td>Description</td>
<td>Collection Frequency</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
<td>------</td>
<td>-------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>ParseFailureCount</td>
<td>Parse Count (Failures)</td>
<td>count</td>
<td>The number of parse failures during the selected time interval. Statistic: Sum Interval: 1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>PhysicalReads</td>
<td>Physical Reads</td>
<td>count</td>
<td>The number of data blocks read from disk, during the selected time interval. Statistic: Sum Interval: 1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>PhysicalReadsTotalBytes</td>
<td>Physical Read Total Bytes</td>
<td>count</td>
<td>The size in bytes of disk reads by all database instance activity including application reads, backup and recovery, and other utilities, during the selected time interval. Statistic: Sum Interval: 1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>PhysicalWrites</td>
<td>Physical Writes</td>
<td>count</td>
<td>The number of data blocks written to disk, during the selected time interval. Statistic: Sum Interval: 1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>PhysicalWriteTotalBytes</td>
<td>Physical Write Total Bytes</td>
<td>count</td>
<td>The size in bytes of all disk writes for the database instance including application activity, backup and recovery, and other utilities, during the selected time interval. Statistic: Sum Interval: 1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>Metric</td>
<td>Metric Display Name</td>
<td>Unit</td>
<td>Description</td>
<td>Collection Frequency</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>QueryLatency</td>
<td>Query Latency</td>
<td>Milliseconds</td>
<td>The time taken to display the results of a simple query on the user's screen.</td>
<td>5 minutes</td>
</tr>
<tr>
<td>QueuedStatements</td>
<td>Queued Statements</td>
<td>count</td>
<td>The number of queued SQL statements, aggregated across all consumer groups, during the selected interval.</td>
<td>NA</td>
</tr>
<tr>
<td>RedoGenerated</td>
<td>Redo Generated</td>
<td>count</td>
<td>Amount of redo generated in bytes, during the selected time interval.</td>
<td>1 minute</td>
</tr>
<tr>
<td>RunningStatements</td>
<td>Running Statements</td>
<td>count</td>
<td>The number of running SQL statements, aggregated across all consumer groups, during the selected interval.</td>
<td>1 minute</td>
</tr>
<tr>
<td>Sessions</td>
<td>Sessions</td>
<td>count</td>
<td>The number of sessions in the database.</td>
<td>1 minute</td>
</tr>
<tr>
<td>SQLNetBytesFromClient</td>
<td>Bytes Received via SQL*Net from Client</td>
<td>count</td>
<td>The number of bytes received from the client over Oracle Net Services, during the selected time interval.</td>
<td>1 minute</td>
</tr>
<tr>
<td>Metric</td>
<td>Metric Display Name</td>
<td>Unit</td>
<td>Description</td>
<td>Collection Frequency</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>SQLNetBytesFromDBLink</td>
<td>Bytes Received via SQL*Net from DBLink</td>
<td>count</td>
<td>The number of bytes received from a database link over Oracle Net Services, during the selected time interval. Statistic: Sum Interval: 1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>SQLNetBytesToClient</td>
<td>Bytes Sent via SQL*Net to Client</td>
<td>count</td>
<td>The number of bytes sent to the client from the foreground processes, during the selected time interval. Statistic: Sum Interval: 1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>SQLNetBytesToDBLink</td>
<td>Bytes Sent via SQL*Net to DBLink</td>
<td>count</td>
<td>The number of bytes sent over a database link, during the selected time interval. Statistic: Sum Interval: 1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>StorageUtilization</td>
<td>Storage Utilization</td>
<td>percent</td>
<td>The percentage of provisioned storage capacity currently in use. Represents the total allocated space for all tablespaces. Statistic: Mean Interval: 1 hour</td>
<td>1 hour</td>
</tr>
<tr>
<td>TransactionCount*</td>
<td>Transaction Count</td>
<td>count</td>
<td>The combined number of user commits and user rollbacks during the selected interval. Statistic: Sum Interval: 1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>Metric</td>
<td>Metric Display Name</td>
<td>Unit</td>
<td>Description</td>
<td>Collection Frequency</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>UserCalls*</td>
<td>User Calls</td>
<td>count</td>
<td>The combined number of logons, parses, and execute calls during the selected interval. Statistic: Sum Interval: 1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>UserCommits</td>
<td>User Commit</td>
<td>count</td>
<td>The number of user commits during the selected time interval. When a user commits a transaction, the generated redo that reflects the changes made to database blocks must be written to disk. Commits often represent the closest thing to a user transaction rate. Statistic: Sum Interval: 1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>UserRollbacks</td>
<td>User Rollbacks</td>
<td>count</td>
<td>Number of times users manually issue the ROLLBACK statement or an error occurs during a user's transactions, during the selected time interval. Statistic: Sum Interval: 1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>WaitTime*</td>
<td>Wait Time</td>
<td>seconds per second</td>
<td>Average rate of accumulation of non-idle wait time by foreground sessions in the database over the time interval. The wait time component of Average Active Sessions. Statistic: Mean Interval: 1 minute</td>
<td>1 minute</td>
</tr>
</tbody>
</table>
Backing Up and Restoring Autonomous Database

This section describes backup and recovery tasks on Autonomous Database.

Topics

- About Backup and Recovery on Autonomous Database
- Restore and Recover your Autonomous Database
- Manual Backups on Autonomous Database

About Backup and Recovery on Autonomous Database

Describes Autonomous Database automatic and manual backups that allow you to restore and recover your database.

Automatic Backups

Autonomous Database automatically backs up your database for you. The retention period for backups is 60 days. You can restore and recover your database to any point-in-time in this retention period.

Autonomous Database performs a full backup every 60 days. Autonomous Database also performs weekly cumulative and daily incremental backups (both displayed as incremental backups).

Manual Backups

You do not have to do any manual backups for your database as Autonomous Database backs up your database automatically. You can perform manual backups from the Oracle Cloud Infrastructure Console; for example if you want to take a backup before a major change to make restore and recovery faster. The retention period for manual backups is 60 days. The manual backups are put in the Oracle Cloud Infrastructure Object Storage bucket you configure for manual backups. When you initiate a point-in-time recovery Autonomous Database decides which backup to use for faster recovery.

Recovery

You can initiate recovery for your database using the Oracle Cloud Infrastructure Console. Autonomous Database automatically restores and recovers your database to the point-in-time you specify or using the backup you select from a list of backups.

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.
Notes for Backup and Recovery

• **Files on Object Store**: For external tables, partitioned external tables, and the external partitions of hybrid partitioned tables, backups do not include the external files that reside on Object Store. Thus, for operations where you use a backup to restore your database, such as Restore or Clone from a backup, it is your responsibility to backup and restore if necessary, the external files associated with external tables, external partitioned tables, or the external files for a hybrid partitioned table.

  See Restore and Recover your Autonomous Database for information on Restore.

  See Clone Autonomous Database from a Backup for information on Clone from a backup.

• **Stopped Database**: Automatic backups occur when a database is stopped. If you stop your database, you do not miss a regularly scheduled automatic backup.

### Restore and Recover your Autonomous Database

From the Oracle Cloud Infrastructure Console you can restore the database using the Restore operation, where you initiate recovery for your database.

Perform the following prerequisite steps as necessary:

• Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.

• From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.

• On the Autonomous Databases page select an Autonomous Database from the links under the Display Name column.

To restore and recover your database, do the following:

1. On the Autonomous Database Details page, from the More Actions drop-down list, select Restore to display the Restore prompt.

2. In the Restore prompt, select Enter Timestamp or Select Backup to restore to a point in time or to restore from a specified backup.
   - Enter Timestamp: Enter a timestamp to restore to in the Enter Timestamp calendar field.
Click the Calendar icon to show the date and timestamp calendar selector. The retention period for backups is 60 days. You can select a timestamp to restore and recover your database to any point-in-time in this retention period.

- Select Backup: Select a backup from the list of backups. Limit the number of backups you see by specifying a period using the From and To calendar fields.
3. **Click** **Restore**.

**Note:**

Restoring Autonomous Database puts the database in the unavailable state during the restore operation. You cannot connect to a database in that state. The only lifecycle management operation supported in unavailable state is **Terminate**.

When concurrent operations such as scaling or creating a manual backup are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

The details page shows Lifecycle State: **Restore In Progress**

4. When the restore operation finishes your Autonomous Database instance opens and the Lifecycle State shows **Available**.

At this point you can connect to your Autonomous Database instance and check your data to validate that the restore point you specified was correct. After checking your
data if you find that the restore date you specified was not the one you needed, you can initiate another restore operation to another point in time.

Notes:

- The restore operation also restores the `DATA_PUMP_DIR` directory and user defined directories to the timestamp you specified for the restore; files that were created after that timestamp would be lost.
- When you restore, the Oracle Machine Learning workspaces, projects, and notebooks are not restored.
- For external tables, partitioned external tables, and the external partitions of hybrid partitioned tables a backup does not include the external files that reside on your Object Store. Thus, for operations where you use a backup to restore your database, such as **Restore** or **Clone from a backup** it is your responsibility to backup, and restore if necessary, the external files associated with external tables, external partitioned tables, or the external files for a hybrid partitioned table.

See [Clone Autonomous Database from a Backup](#) for information on using **Clone from a backup**.

---

**Manual Backups on Autonomous Database**

In addition to automatic backups that Autonomous Database regularly performs, you can take manual backups at any time.

Topics

- Configure Manual Backups on Autonomous Database
- Perform Manual Backups on Autonomous Database
- Manual Backup Notes

**Configure Manual Backups on Autonomous Database**

Follow these steps to configure the Oracle Cloud Infrastructure Object Storage credentials and tenancy URL for manual backups and to create the Oracle Cloud Infrastructure Object Storage bucket to store manual backups.

To configure your database for manual backups, do the following:

- Create an Oracle Cloud Infrastructure Object Storage bucket for manual backups
- Set database property `DEFAULT_BACKUP_BUCKET` to specify the manual backup bucket on the Oracle Cloud Infrastructure Object Storage
- Define the Oracle Cloud Infrastructure Object Storage credentials
- Set the database property `DEFAULT_CREDENTIAL` database property

These 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 the bucket change.
To perform the manual backup configuration tasks you need to connect to your database and perform the required DDL commands. For example, connect to your database with SQL Developer or SQL*Plus. See Connect to Autonomous Database Using Oracle Database Tools.

1. Create an Oracle Cloud Infrastructure Object Storage bucket for manual backups for your database. Specify the bucket name to meet your manual backup needs.
   a. Open the Oracle Cloud Infrastructure Console.
   b. Select **Storage** from the menu.
   c. Under Storage, select **Object Storage and Archive Storage**.
   d. Click **Create Bucket**.
      This displays the Create Bucket dialog.
   e. In the Create Bucket page, enter the **Bucket Name** and click **Create**.

   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.

   The format of the URL to specify a bucket is:

   \[https://swiftobjectstorage.region.oraclecloud.com/v1/namespace-string/bucket_name\]

   For example, if you provision an Autonomous Database named `DB2020AA`, then the bucket name that you create could be `DB2020AABAK` (you can use any valid bucket name). Following the same example, the URL of this bucket in the `us-phoenix-1` region is:

   \[https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/namespace-string/DB2020AABAK\]

   Where `namespace-string` is the Oracle Cloud Infrastructure object storage namespace. See Understanding Object Storage Namespaces for more information. See Regions and Availability Domains for a list of regions.

2. Set the database property **DEFAULT_BACKUP_BUCKET** to identify the manual backup bucket in your Oracle Cloud Infrastructure Object Storage. You need to perform this step as the ADMIN user.
   • Set the database property **DEFAULT_BACKUP_BUCKET** to specify the manual backup bucket URL.

   For example:

   ```sql
   ALTER DATABASE PROPERTY SET default_backup_bucket='https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/namespace-string/bucket_name';
   ```

   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`.

Note that you need to run this command as the ADMIN user. For example:

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

See `CREATE_CREDENTIAL Procedure` for details on the arguments for `username` and `password`.

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';
```

Use the following command to show the value for `DEFAULT_BACKUP_BUCKET`:

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

See `Manual Backup Notes` for more information.

### Perform Manual Backups on Autonomous Database

In addition to automatic backups, Autonomous Database also allows you to take manual backups and store the backups to Oracle Cloud Infrastructure Object Storage.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the ⌧ next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click `Oracle Database` and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomic Databases page select an Autonomous Database from the links under the `Display Name` column.

1. On the Autonomous Database Details page, under `Resources`, click `Backups`.
2. On the Autonomous Database Details page, under `Backups`, click `Create Manual Backup`.
3. If you previously configured your database for manual backups, continue to Step 4.

The system checks and shows a dialog box if the database is not configured for manual backups. In this case, to perform a manual backup click Close and configure your database for manual backups. See Configure Manual Backups on Autonomous Database for more information.

4. In the Create Manual Backup dialog box enter a name in the Display Name field.

5. In the Create Manual Backup dialog box click Create Manual Backup.

See Manual Backup Notes for more information.

Manual Backup Notes

Provides notes for manual backups and for manual backup configuration.

Notes for manual backup configuration:

- If you previously configured manual backups using the DEFAULT_BUCKET property, you do not need to make any changes to perform manual backups with your existing configuration. In this case, the DEFAULT_BUCKET property is set to the value of Oracle Cloud Infrastructure Object Storage tenancy URL and the required bucket name is in the format of backup_databasesname. Where databasesname is
lowercase. However, Oracle recommends that you configure Autonomous Database for manual backups using the `DEFAULT_BACKUP_BUCKET` database property. See Configure Manual Backups on Autonomous Database for more information.

- If you previously configured Autonomous Database to use manual backups using the `DEFAULT_BUCKET` property and created backups, then after configuring the `DEFAULT_BACKUP_BUCKET` property to use a new manual backup bucket, the old manual backups in the old bucket are not available for restore. If you want to use the old backups then you must change the value of the `DEFAULT_BACKUP_BUCKET` property to specify the URL of the old manual backup bucket.

- If you previously configured Autonomous Database to use manual backups and you rename your Autonomous Database, then your backups will continue to work without changes.

Notes for manual backups:

- 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 Database instance when you initiate a point-in-time-recovery.

- The retention period for manual backups is the same as for automatic backups, which is 60 days.

- While backing up a database, the database is fully functional. However, during the backup the lifecycle management operations, such as stopping the database, are not allowed.
Cloning and Moving an Autonomous Database

Autonomous Database provides cloning where you can choose to create a full clone, create a metadata clone, or create a refreshable clone. You can also use cloning to upgrade your database to a newer Oracle Database version when newer Oracle Database versions are available. You can move an Autonomous Database to a different Oracle Cloud Infrastructure compartment.

When you create a clone for an Autonomous Database instance, you have the option to select the clone type:

- **Full Clone**: creates a new database with the source database's data and metadata.
- **Refreshable Clone**: creates a read-only full clone that can be easily refreshed with the data from the source database. See [Using Refreshable Clones with Autonomous Database](#) for more information.
- **Metadata Clone**: creates a new database that includes all of the source database schema metadata, but not the source database data.

For a Full Clone or a Metadata Clone, you have the option to select the clone source:

- Clone from a database instance. This creates a clone of a running database.
- Clone from a backup. This creates a clone when you select a backup from a list of backups, or when you enter a backup point-in-time to clone.

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 Database Instance](#)
- [Clone Autonomous Database from a Backup](#)
- [Notes for Cloning Autonomous Database](#)
- [Move an Autonomous Database to a Different Compartment](#)

**Clone an Autonomous Database Instance**

Shows you the steps to clone a database from the Oracle Cloud Infrastructure Console.

Perform the following prerequisite steps as necessary:

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

• Choose your region. See **Switching Regions** for information on switching regions and working in multiple regions.

• Choose your **Compartment**. See **Compartments** for information on using and managing compartments.

• Select an Autonomous Database instance from the list in your compartment.

To clone the Autonomous Database instance, do the following:

1. On the **Autonomous Database Details** page, from the **More Actions** drop-down list, select **Create Clone**.

2. On the **Create Autonomous Database Clone** page, choose the clone type from the choices:
   - **Full Clone**: creates a new database with the source database's data and metadata.
   - **Refreshable Clone**: creates a read-only full clone that can be easily refreshed with the source database's data. See **Using Refreshable Clones with Autonomous Database** for more information.
   - **Metadata Clone**: creates a new database with the source database's metadata without the data.

3. In the **Clone source** area, select one of:
   - **Clone from database instance**: This creates a clone from a running database.
   - **Clone from a backup**: This creates a database clone using a selected backup or from a point-in-time timestamp that you enter. See **Clone Autonomous Database from a Backup** for more information.

4. Provide basic information for the Autonomous Database clone.
   - **Choose your preferred region**: from the list, select the region where you want to create the clone.
     
     Note: the list only shows the regions that you are subscribed to.
     
     When you select **Clone from database instance** as the clone source, and you choose another region, other than the current region for your clone target, you can only perform such a cross region clone using the Oracle Cloud Infrastructure Console. Using either the Oracle Cloud Infrastructure CLI or Terraform is not supported for creating a cross-region clone.
   - **Create in Compartment**. See **Compartments** for information on using and managing compartments.
     
     Note: the list only shows the compartments that you are subscribed to.
   - **Display name** Specify a user-friendly description or other information that helps you easily identify the resource.
     
     You can use the name provided, of the form: **Clone-of-DBname** or change this to the name you want to use to identify the database. The supplied **DBname** is the name of the source database that you are cloning.
4. **Database name**: Specify the database name; it must consist of letters and numbers only. The maximum length is 30 characters. The same database name cannot be used for multiple Autonomous Databases in the same tenancy in the same region.

   The default database name is a generated 16-character string.

5. 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 database.
     
     Your license type determines the **OCPU count** maximum. For example, if your license type is Bring Your Own License (BYOL) with Oracle Database Standard Edition (SE), the **OCPU count** maximum is 8.
   - **OCPU auto scaling** By default OCPU auto scaling is enabled to allow the system to automatically use up to three times more CPU and IO resources to meet workload demand. If you do not want to use OCPU auto scaling then deselect this option.
     
     See Use Auto Scaling for more information.
   - **Storage (TB)** Specify the storage you wish to make available to your 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.
   - **Storage auto scaling** By default storage auto scaling is disabled. Select if you want to enable storage auto scaling to allow the system to automatically expand to use up to three times more storage.
     
     See Use Auto Scaling for more information.

6. Create administrator credentials.

   - **Username** This is a read-only field.
   - **Password** Set the password for the Autonomous Database Admin user. The password must meet the strong password complexity criteria based on Oracle Cloud security standards. For more information on the password complexity rules see Create Users on Autonomous Database - Connecting with a Client Tool.
     
     - **Confirm password** Specify a value to confirm the password.

7. Choose network access

   - **Secure access from everywhere** By default all secure connections are allowed from everywhere.
   - **Secure access from allowed IPs and VCNs only** This option restricts connections to the database according to the access control lists (ACLs) you specify. To add multiple ACLs for the Autonomous Database, click + Access Control Rule.
See Configure Access Control Lists When You Provision or Clone an Instance for more information.

- **Private endpoint access only**
  This option assigns a private endpoint, private IP, and hostname to your database. Specifying this option allows traffic only from the VCN you specify; access to the database from all public IPs or VCNs is blocked. This allows you to define security rules, ingress/egress, at the Network Security Group (NSG) level and to control traffic to your Autonomous Database.

  See Configure Private Endpoints When You Provision or Clone an Instance for more information.

8. **Choose license and Oracle Database Edition**

- **Bring Your Own License (BYOL)**
  Select if your organization already owns Oracle database software licenses. Bring your existing database software licenses to the database cloud service. See Cloud pricing for information on Bring Your Own License (BYOL) and other licensing options for Oracle Cloud Infrastructure cloud service pricing.

- **Choose an Oracle Database Edition**
  When you select Bring Your Own License (BYOL), you also choose an Oracle Database Edition. The Oracle Database Edition you select is based on the license you bring to Autonomous Database and changes the maximum value that you can select for the OCPU count. The choices are:

  **Oracle Database Enterprise Edition (EE):** For this license type the maximum allowed value for OCPU count is 128, however you may contact your Oracle account team to request more than 128 OCPUs. With auto scaling enabled you can use up to OCPU count x 3 OCPUs. For example, if you set the OCPU count to 128, you can use up to 384 OCPUs.

  **Oracle Database Standard Edition (SE):** For this license type the maximum allowed value for OCPU count is 8. With auto scaling enabled you can use up to OCPU count x 3 OCPUs. This license restricts the number of OCPUs you can use to a maximum of 8 OCPUs, with or without auto scaling enabled.

  See Use Auto Scaling for more information.

- **License Included**
  Subscribe to new database software licenses and the database cloud service.

9. **(Optional) Provide up to 10 maintenance contacts.**

  Click Add Contact and in the Contact Email field, enter a valid email address. If the database you are cloning has a customer contact list, the list is copied. To enter multiple Contact Email addresses, repeat the process to add up to 10 customer contact emails.

  See View and Manage Customer Contacts for Operational Issues and Announcements for more information.

10. **(Optional) Click Show Advanced Options to select advanced options.**

- **Encryption Key**
  **Encryption using Oracle-managed keys:** By default Autonomous Database uses Oracle-managed encryption keys. Using Oracle-managed keys, Autonomous Database creates and manages the encryption keys that protect your data and Oracle handles rotation of the TDE master key.
Encrypt using customer-managed keys: If you select customer-managed keys, a master encryption key in the Oracle Cloud Infrastructure Vault is used to generate the TDE master key on Autonomous Database.

See Use Customer-Managed Encryption Keys on Autonomous Database for more information.

- **Maintenance**
  - **Patch level** By default the patch level is the patch level of the source database. Select Early to configure the instance with the early patch level. When cloning a source database with Early patch level, you can only choose the Early patch level for your clone.

  See Set the Patch Level for more information.

- **Management**
  Shows the character set and national character set for your database.

  See Choose a Character Set for Autonomous Database for more information.

- **Tags**
  If you want to use Tags, enter the TAG KEY and VALUE. Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values which can be attached to resources.

  See Tagging Overview for more information.

11. Click **Create Autonomous Database Clone**.

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

If you created a cross-region clone, a new tab shows the newly provisioned clone's Oracle Cloud Infrastructure Console and the region shown is the region you selected when you created the clone.

See Notes for Cloning Autonomous Database for additional information on cloning.

See Cloning an Autonomous Database for information on using the API.

## Clone Autonomous Database from a Backup

Shows the options to select a backup as the clone source for cloning Autonomous Database.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.

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

1. Choose your region. See Switching Regions for information on switching regions and working in multiple regions.

2. Choose your **Compartment**. See Compartments for information on using and managing compartments.

3. Select an Autonomous Database instance from the list in your compartment.

4. On the **Details** page, from the **More Actions** drop-down list, select **Create Clone**.
5. On the On the **Create Autonomous Database Clone** page, choose the clone type from the choices:
   - **Full Clone**: creates a new database with the source database's data and metadata.
   - **Refreshable Clone**: creates a read-only full clone that can be easily refreshed with the source database's data. See [Using Refreshable Clones with Autonomous Database](#) for more information.
   - **Metadata Clone**: creates a new database with the source database's metadata without the data.

6. In the **Configure clone source** area, select the **Clone source** option:
   - **Clone from database instance**: This creates a clone from a running database. See [Clone an Autonomous Database Instance](#) for details and steps with this selection.
   - **Clone from a backup**: This selection creates a database clone using a backup 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**
8. 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 30 characters. The same database name cannot be used for multiple Autonomous Databases in the same tenancy in the same region.
     The default database name is a generated 16-character string.

9. 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 database.
     Your license type determines the OCPU count maximum. For example, if your license type is Bring Your Own License (BYOL) with Oracle Database Standard Edition (SE), the OCPU count maximum is 8.
   - **OCPU auto scaling** By default OCPU auto scaling is enabled to allow the system to automatically use up to three times more CPU and IO resources to meet workload demand. If you do not want to use OCPU auto scaling then deselect this option.
See Use Auto Scaling for more information.

- **Storage (TB)** Specify the storage you wish to make available to your 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.

- **Storage auto scaling** By default storage auto scaling is disabled. Select if you want to enable storage auto scaling to allow the system to automatically expand to use up to three times more storage.
  
  See Use Auto Scaling for more information.

10. Create administrator credentials.

- **Username** This is a read-only field.

- **Password** Set the password for the Autonomous Database Admin user. The password must meet the strong password complexity criteria based on Oracle Cloud security standards. For more information on the password complexity rules see Create Users on Autonomous Database - Connecting with a Client Tool.

- **Confirm password** Specify a value to confirm the password.

11. Choose network access

**Note:**

After you clone your Autonomous Database you can change the network access option you select for the cloned instance.

- **Secure access from everywhere**
  
  By default all secure connections are allowed from everywhere.

- **Secure access from allowed IPs and VCNs only**
  
  This option restricts connections to the database according to the access control lists (ACLs) you specify. To add multiple ACLs for the Autonomous Database, click + Access Control Rule.
  
  See Configure Access Control Lists When You Provision or Clone an Instance for more information.

- **Private endpoint access only**
  
  This option assigns a private endpoint, private IP, and hostname to your database. Specifying this option allows traffic only from the VCN you specify; access to the database from all public IPs or VCNs is blocked. This allows you to define security rules, ingress/egress, at the Network Security Group (NSG) level and to control traffic to your Autonomous Database.
  
  See Configure Private Endpoints When You Provision or Clone an Instance for more information.

12. Choose license and Oracle Database Edition

- **Bring Your Own License (BYOL)**
  
  Select if your organization already owns Oracle database software licenses. Bring your existing database software licenses to the database cloud service.
See Cloud pricing for information on Bring Your Own License (BYOL) and other licensing options for Oracle Cloud Infrastructure cloud service pricing.

• Choose an Oracle Database Edition

When you select Bring Your Own License (BYOL), you also choose an Oracle Database Edition. The Oracle Database Edition you select is based on the license you bring to Autonomous Database and changes the maximum value that you can select for the OCPU count. The choices are:

Oracle Database Enterprise Edition (EE): For this license type the maximum allowed value for OCPU count is 128, however you may contact your Oracle account team to request more than 128 OCPUs. With auto scaling enabled you can use up to OCPU count x 3 OCPUs. For example, if you set the OCPU count to 128, you can use up to 384 OCPUs.

Oracle Database Standard Edition (SE): For this license type the maximum allowed value for OCPU count is 8. With auto scaling enabled you can use up to OCPU count x 3 OCPUs. This license restricts the number of OCPUs you can use to a maximum of 8 OCPUs, with or without auto scaling enabled.

See Use Auto Scaling for more information.

• License Included

Subscribe to new database software licenses and the database cloud service.

13. (Optional) Provide up to 10 maintenance contacts.

Click Add Contact and in the Contact Email field, enter a valid email address. If the database you are cloning has a customer contact list, the list is copied. To enter multiple Contact Email addresses, repeat the process to add up to 10 customer contact emails.

See View and Manage Customer Contacts for Operational Issues and Announcements for more information.

14. (Optional) Click Show Advanced Options to select advanced options.

• Encryption Key

Encryption using Oracle-managed keys: By default Autonomous Database uses Oracle-managed encryption keys. Using Oracle-managed keys, Autonomous Database creates and manages the encryption keys that protect your data and Oracle handles rotation of the TDE master key.

Encrypt using customer-managed keys: If you select customer-managed keys, a master encryption key in the Oracle Cloud Infrastructure Vault is used to generate the TDE master key on Autonomous Database.

See Use Customer-Managed Encryption Keys on Autonomous Database for more information.

• Maintenance

Patch level By default the patch level is the patch level of the source database. Select Early to configure the instance with the early patch level. When cloning a source database with Early patch level, you can only choose the Early patch level for your clone.

See Set the Patch Level for more information.

• Management

Shows the character set and national character set for your database.

See Choose a Character Set for Autonomous Database for more information.

• Tags
If you want to use Tags, enter the **TAG KEY** and **VALUE**. Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values which can be attached to resources.

See [Tagging Overview](#) for more information.

15. Click **Create Autonomous Database Clone**.

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

---

**Notes:**

- If there is an ongoing clone from backup operation on a source database, you cannot initiate a new clone operation on the same backup being cloned until the ongoing operation completes. Thus, you cannot clone from backup twice concurrently from a specific backup (for example, a specific timestamp or a specific selected backup from the list of backups).

- For external tables, partitioned external tables, and the external partitions of hybrid partitioned tables a backup does not include the external files that reside on your Object Store. Thus, for the clone from backup operation, it is your responsibility to backup, and restore if necessary, the external files associated with external tables, external partitioned tables, or the external files for a hybrid partitioned table.

- With clone from backup, the Oracle Machine Learning workspaces, projects, and notebooks of the source database are not cloned to the new database.

See [Notes for Cloning Autonomous Database](#) for additional information on cloning.

See [Cloning an Autonomous Database](#) for information on using the API.

---

### Notes for Cloning Autonomous Database

Provides information about the cloning operation and the resulting cloned 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 of the time of the backup).

- If you create a clone and the source database has an access control list (ACL) and you specify the private endpoint network access option, **Virtual cloud network** for the target database, the ACL is not cloned to the new database. In this case, you must define security rules within your Network Security Group (or groups) to control traffic to and from your target database (instead of using the access control rules that were specified in the ACL on the clone source). See [Configure Private Endpoints When You Provision or Clone an Instance](#) for more information.
• For a **Metadata Clone**, the APEX Apps and the OML Projects and Notebooks are copied to the clone. For a **Metadata Clone**, the underlying database data of the APEX App or OML Notebook is not cloned.

• The Autonomous Database Details page for an Autonomous Database instance that was created by cloning includes the **Cloned From** field. This displays the name of the database where the clone was created.

• When you select **Clone from backup** as the clone source, you can only clone an Autonomous Database instance to the same tenancy and the same region as the source database.

• When you select **Clone from database instance** as the clone source, and you choose another region, other than the current region for your clone target, you can only perform such a cross region clone using the Oracle Cloud Infrastructure Console. Using either the Oracle Cloud Infrastructure CLI or Terraform is not supported for creating a cross-region clone.

### 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 in the Database Monitor card (under **Monitor** in Database Actions) for the cloned database.

For more information on setting resource management rules, see Manage Runaway SQL Statements on Autonomous Database.

### Optimizer Statistics for a Cloned Database

During the provisioning for either a Full Clone or a Metadata Clone, the optimizer statistics are copied from the source database to the cloned database.

The following applies for optimizer statistics for tables in a cloned database:

• **Full Clone**: loads into tables behave the same as loading into a table with statistics already in place.

• **Metadata Clone**: the first load into a table after the 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 Database to a Different Compartment

Shows you the steps to move a database to a different Oracle Cloud Infrastructure compartment.

**Note:**
- To move a database you must have the right to manage Autonomous Databases in the database's current compartment and in the compartment you are moving it to.
- As soon as you move a database to a different compartment, the policies that govern the new compartment apply immediately and affect access to the database. Therefore, your access to the database may change, depending on the policies governing your Oracle Cloud user account's access to resources.

Perform the following prerequisite steps as necessary:

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

1. Choose your region. See Switching Regions for information on switching regions and working in multiple regions.
2. Choose your Compartment. See Compartments for information on using and managing compartments.
3. Select an Autonomous Database instance from the list in your compartment.
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.
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
- View and Update Your License and Oracle Database Edition 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 that a cloud account owns in a region.

Topics

- About Wallet Rotation
- Rotate Wallets with Immediate Rotation
- Rotate Wallets with Grace Period

About Wallet Rotation

You have the option to perform one of two types of wallet rotation: immediate or with a grace period.

- **Immediate** wallet rotation initiates immediately, without delay.
- **After a grace period** wallet rotation occurs with a grace period. During the grace period the old client certification keys remain valid for a selected time of 1 hour to 24 hours. After the grace period expires only the new client certification keys are valid.

You may want to rotate wallets for any of the following reasons:

- If your organization's policies require regular client certification key rotation.
- When a client certification key or a set of keys is suspected to be compromised.
Rotate Wallets with Immediate Rotation

Immediate wallet rotation lets you invalidate existing client certification keys for an Autonomous Database instance or for all Autonomous Database instances that a cloud account owns in a region.

There are two options for immediate client certification key rotation:

• Per-database with **Instance Wallet** selected:
  – For the database whose certification key is rotated, any existing database specific instance wallets will be void. After you rotate a wallet you have to download a new wallet to connect to the database.
  – Regional wallets containing all database certification keys continue to work.
  – All user sessions are terminated for the database whose wallet is rotated. User session termination begins after wallet rotation completes, however this process does not happen immediately.

  **Note:**
  If you want to terminate all connections immediately after the wallet rotation completes, Oracle recommends that you restart the Autonomous Database instance. This provides the highest level of security for your database.

• Regional level with **Regional Wallet** selected:
  – For the region whose certification key is rotated, both regional and database specific instance wallets will be void. After you rotate a wallet you have to download new regional or instance wallets to connect to any database in the region.
  – All user sessions are terminated for the databases in the region whose wallet is rotated. User session termination begins after wallet rotation completes, however this process does not happen immediately.

  **Note:**
  If you want to terminate all connections immediately after the wallet rotation completes, Oracle recommends that you restart the Autonomous Database instances in the region. This provides the highest level of security for your database.

To immediately rotate the client certification key for a given database or for all Autonomous Database instances that a cloud account owns in a region:

1. Navigate to the Autonomous Database details page.
2. Click **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.

Oracle recommends you provide a database-specific instance wallet to end users and for application use whenever possible, with Wallet Type set to **Instance Wallet** when you use **Download Wallet**. Regional wallets should only be used for administrative purposes that require potential access to all Autonomous Databases within a region.

You can also use the Autonomous Database API to rotate wallets using **UpdateAutonomousDatabaseRegionalWallet** and **UpdateAutonomousDatabaseWallet**. See **Autonomous Database Wallet Reference** for more information.

---

**Rotate Wallets with Grace Period**

Autonomous Database allows you to rotate wallets for an Autonomous Database instance or for all instances that a cloud account owns in a region, with a grace period of 1 hour to 24 hours.

Setting a grace period allows you to perform wallet rotation without down time. During the grace period you can inform users to download the new wallet and to update their applications to use the new wallet. During the grace period both the old and new client certification keys are valid. When the grace period expires, Autonomous Database invalidates the old client certification keys and only the new client certification keys are valid.

There are two options for client certification key rotation with a grace period:

• **Per-database with Instance Wallet** selected:
  – For the database whose certification key is rotated, database specific instance wallets that were in use before the wallet rotation be void after the grace period expires.
  – After you perform client certification key rotation with a grace period, you can immediately download a wallet and use the new wallet to connect to the database.
  – Regional wallets containing all database certification keys continue to work.
  – After the grace period expires, existing connections using the old wallet continue to work.

  **Note:**
  After the grace period completes, if you want to terminate any connections using the old wallet, Oracle recommends that you restart the Autonomous Database instance.

• **Regional level with Regional Wallet** selected:
– For the region whose certification key is rotated, both regional and database specific instance wallets will be void. After the grace period expires you have to download new regional or instance wallets to connect to any database in the region.

– After the grace period expires, existing connections using the old wallets continue to work.

**Note:**

After the grace period completes, if you want to terminate any connections using the old wallets, Oracle recommends that you restart every Autonomous Database instance in the region.

To rotate the client certification key with a grace period for a given database or for all Autonomous Database instances that a cloud account owns 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 in the region that a cloud account owns).
4. Click **Rotate Wallet**.
5. Select **After a grace period**.
6. In the **Grace period (in hours)** area, either enter a value in the text field or use the slider to select a value.
7. Enter the name as shown in the dialog to confirm the wallet rotation.

8. In the Rotate Wallet dialog, click **Rotate**.

The Database Connection page shows: **Rotation in Progress**.

After the rotation completes, the **Wallet last rotated** field shows the last rotation date and time.

Notes for wallet rotation with a grace period:

- Always Free Autonomous Databases only support immediate wallet rotation (wallet rotation with a grace period is not supported).

- Oracle recommends you provide a database-specific instance wallet to end users and for application use whenever possible, with Wallet Type set to **Instance Wallet** when you use **Download Wallet**. Regional wallets should only be used for administrative purposes that require potential access to all Autonomous Databases within a region.
You can also use the Autonomous Database API to rotate wallets using UpdateAutonomousDatabaseRegionalWallet and UpdateAutonomousDatabaseWallet. See Autonomous Database Wallet Reference for more information.

View and Update Your License and Oracle Database Edition on Autonomous Database

Describes how to view and update your license type and Oracle Database Edition for Autonomous Database.

The License Type field on the Autonomous Database Information tab shows your license and Oracle Database Edition. For example:

License Type: Bring Your Own License (BYOL), Enterprise Edition

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the Display Name column.

To change your license type or Oracle Database Edition:

2. On the Update License and Oracle Database Edition page select a license and Oracle Database Edition:
   - Bring Your Own License (BYOL) Select if your organization already owns Oracle database software licenses. Bring your existing database software licenses to the database cloud service.
   - Choose an Oracle Database Edition:
     - Oracle Database Enterprise Edition (EE): For this license type the maximum allowed value for OCPU count is 128, however you may contact your Oracle account team to request more than 128 OCPUs. With auto scaling enabled you can use up to OCPU count x 3 OCPUs. For example, if you set the OCPU count to 128, you can use up to 384 OCPUs.
     - Oracle Database Standard Edition (SE): For this license type the maximum allowed value for OCPU count is 8. With auto scaling enabled you can use up to OCPU count x 3 OCPUs. This license restricts the number of OCPUs you can use to a maximum of 8 OCPUs, with or without auto scaling enabled.

The edition you select is based on the license you bring to Autonomous Database and changes the maximum value that you can select for the OCPU count.

See Use Auto Scaling for more information.
• **License Included**

Subscribe to new database software licenses and the database cloud service.

3. Click **Save Changes**.

While the system applies the changes, the lifecycle state changes to **Updating**. When the operation completes the lifecycle state shows **Available**.

If you have not specified a complete license type, Autonomous Database instances created before the introduction of the Oracle Database Edition will see the following message on the Oracle Cloud Infrastructure Console:

![Message](image.png)

To dismiss this message, click **Update License and Oracle Database Edition** and set a license type and Oracle Database Edition.

For information on Bring Your Own License (BYOL) and other licensing options for Oracle Cloud Infrastructure, see the following:

• Lower your TCO for Oracle Database Standard Edition with the Bring Your Own License (BYOL) program

• BYOL policies are detailed in: Oracle PaaS and IaaS Universal Credits Service Descriptions

• Frequently asked questions: Oracle Bring Your Own License (BYOL)

• Cloud pricing

**Notes for performing Update License and Oracle Database Edition operation:**

• If you try to switch to License Type: Bring Your Own License (BYOL) Standard Edition when the base OCPU count is above 8, you see the following message:

You cannot select Oracle Database Standard Edition unless your OCPU count is a total of eight (8) or less OCPUs. Use the Manage Scaling option to adjust your OCPU count before changing to Standard Edition.

In this case, lower your OCPU count before you change the license type. See **Remove CPU or Storage Resources or Disable Auto Scaling** for more information.

• If you switch to License Type: Bring Your Own License (BYOL) Standard Edition when the base OCPU count is 8 and OCPU auto scaling is enabled, you see the following message:

OCPU auto scaling will be disabled with a base OCPU count of 8, under an Oracle Database Standard Edition license.

In this case, OCPU auto scaling is disabled and the OCPU count is set to 8.

• If you switch to License Type: Bring Your Own License (BYOL) Standard Edition when the base OCPU count is below 8 and above 2, with OCPU auto scaling enabled, you see the following message:

Your OCPU auto scaling maximum will be set to 8 OCPUs, under an Oracle Database Standard Edition license.

In this case, the maximum number of OCPUs with OCPU auto scaling is 8.
Update Always Free Instance to Paid with Autonomous Database

Describes how to update your instance to paid from free with Autonomous Database.

You can upgrade from an Always Free Autonomous Database account to a paid account at any time. If you have a free Oracle Cloud Infrastructure account this is a two step process:

First, upgrade to a paid Oracle Cloud Infrastructure account.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Always Free Autonomous Database instance from the links under the Display Name column.

**Note:**

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

Upgrade your Always Free Autonomous Database instance as follows:

1. On the Details page, from the More Actions drop-down list, select Update Instance to Paid.
2. On the Update Instance page confirm and proceed with the upgrade.

Safeguard Your Data with Data Safe on Autonomous Database

Oracle Data Safe provides features that help you protect sensitive and regulated data in your database.

See Oracle Data Safe Overview for an overview of Oracle Data Safe.

See Oracle Cloud Infrastructure Regions for information on enabling Oracle Data Safe.

Use Oracle Database Vault with Autonomous Database

Oracle Database Vault implements powerful security controls for your database. These unique security controls restrict access to application data by privileged database
users, reducing the risk of insider and outside threats and addressing common compliance
requirements.

See What Is Oracle Database Vault? for more information.

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 \texttt{DV\_OWNER} and \texttt{DV\_ACCTMGR} roles. If you want to set up
separate users for \texttt{DV\_OWNER} and \texttt{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 \texttt{CREATE} | \texttt{ALTER} | \texttt{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 \texttt{GGADMIN} user \texttt{DV\_GOLDENGATE\_ADMIN} and
  \texttt{DV\_GOLDENGATE\_REDO\_ACCESS}.
- The ADMIN user must grant the \texttt{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 \texttt{DBMS\_MACADM\_AUTHORIZE\_DATAPUMP\_USER}.
  See \texttt{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:

   \texttt{EXEC DBMS\_CLOUD\_MACADM\_CONFIGURE\_DATABASE\_VAULT('adb\_dbv\_owner',
   'adb\_dbv\_acctmgr');}

   Where:
• *adb_dbv_owner* is the Oracle Database Vault owner.
• *adb_dbv_acctmgr* is the account manager.

See **CONFIGURE_DATABASE_VAULT Procedure** for more information.

2. Enable Oracle Database Vault:

```
EXEC DBMS_CLOUD_MACADM.ENABLE_DATABASE_VAULT;
```

See **ENABLE_DATABASE_VAULT Procedure** for more information.

3. Restart the Autonomous Database instance.

See **Restart Autonomous Database** for more information.

Use the following command to check if Oracle Database Vault is enabled or disabled:

```
SELECT * FROM DBA_DV_STATUS;
```

Output similar to the following appears:

```
<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>TRUE</td>
</tr>
</tbody>
</table>
```

The **DV_ENABLE_STATUS** value **TRUE** indicates Oracle Database Vault is enabled.

**Note:**

Autonomous Database maintenance operations such as backups and patching are not affected when Oracle Database Vault is enabled.

See **Disable Oracle Database Vault on Autonomous Database** for information on disabling Oracle Database Vault.

**Disable Oracle Database Vault on Autonomous Database**

Shows the steps to disable Oracle Database Vault on Autonomous Database.

To disable Oracle Database Vault on Autonomous Database, do the following:

1. Disable Oracle Database Vault.

```
EXEC DBMS_CLOUD_MACADM.DISABLE_DATABASE_VAULT;
```

See **DISABLE_DATABASE_VAULT Procedure** for more information.

2. Restart the Autonomous Database instance.

See **Restart Autonomous Database** for more information.
Use the following command to check if Oracle Database Vault is enabled or disabled:

```
SELECT * FROM DBA_DV_STATUS;
```

Output similar to the following appears:

```
NAME                 STATUS
--------------------  -----------
DV_CONFIGURE_STATUS  TRUE
DV_ENABLE_STATUS     FALSE
```

The `DV_ENABLE_STATUS` value `FALSE` indicates Oracle Database Vault is enabled.

Disable User Management with Oracle Database Vault on Autonomous Database

Shows how to disallow user management related operations for specified components on Autonomous Database with Oracle Database Vault enabled.

Autonomous Database with Oracle Database Vault enabled has user management, by default, enabled for the Oracle APEX(APEX) console. If you want to enforce stricter separation of duty and disallow user management from this console, use `DBMS_CLOUD_MACADM.DISABLE_USERMGMT_DATABASE_VAULT`.

1. As a user granted `DV_ACCTMGR` and `DV_ADMIN` roles you can disable user management for specified components.

2. To disable user management for a specified component, for example for the APEX component, use the following command:

   ```
   EXEC DBMS_CLOUD_MACADM.DISABLE_USERMGMT_DATABASE_VAULT('APEX');
   ```

See `DISABLE_USERMGMT_DATABASE_VAULT Procedure` for more information.

Enable User Management with Oracle Database Vault on Autonomous Database

Shows the steps to allow user management for a specified component on Autonomous Database with Oracle Database Vault enabled.

Autonomous Database with Oracle Database Vault enabled has user management, by default, enabled for the Oracle APEX(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.
Managing Encryption Keys on Autonomous Database

Describes how to use customer-managed encryption keys with Autonomous Database, and if you are using customer-managed encryption keys, shows how to rotate the keys, switch to Oracle-managed encryption keys, or view the encryption key history.

Topics

- About Master Encryption Key Management on Autonomous Database
- Prerequisites to Use Customer-Managed Encryption Keys on Autonomous Database
- Use Customer-Managed Encryption Keys on Autonomous Database
- Switch to Oracle-Managed Encryption Keys on Autonomous Database
- View History for Customer-Managed Encryption Keys on Autonomous Database
- Notes for Using Customer-Managed Keys with Autonomous Database

About Master Encryption Key Management on Autonomous Database

Autonomous Database provides two options for Transparent Data Encryption (TDE) to encrypt your database:

- Oracle-managed encryption keys
- Customer-managed encryption keys

Autonomous Database uses Transparent Data Encryption, including a TDE master key and TDE tablespace keys to encrypt data in the database. As shown in the following figure, the TDE master key generates and encrypts/decrypts the TDE tablespace keys, and the TDE tablespace keys encrypt the data files.
Oracle-Managed Master Encryption Keys on Autonomous Database

By default Autonomous Database uses Oracle-managed encryption keys.

Using Oracle-managed keys, Autonomous Database creates and manages the encryption keys that protect your data and Oracle handles rotation of the TDE master key.

Customer-Managed Encryption Keys on Autonomous Database

If your organization's security policies require customer-managed encryption keys, you can configure Autonomous Database to use an Oracle Cloud Infrastructure Vault master encryption key. With customer-managed master encryption keys, Autonomous Database uses the master encryption key to generate the TDE master key, and the TDE master key is encrypted using the Oracle Cloud Infrastructure Vault master key and stored locally on the database.

⚠️ Caution:

The customer-managed encryption key is stored in Oracle Cloud Infrastructure Vault, external to the database host. If the customer-managed encryption key is disabled or deleted, the database will be inaccessible.

Use customer-managed encryption keys by performing the following steps:

1. Create a master encryption key in your Oracle Cloud Infrastructure Vault.

   See Prerequisites to Use Customer-Managed Encryption Keys on Autonomous Database for more information.
2. Select customer-managed encryption keys from the Oracle Cloud Infrastructure Console:

   • For an existing database, select **Manage Encryption Key** on the Oracle Cloud Infrastructure Console.
     
     See [Use Customer-Managed Encryption Keys on Autonomous Database](#) for details.
   
   • While provisioning, under **Advanced Options**, on the **Encryption Key** tab select **Encrypt using customer-managed keys**.
   
   • While cloning, under **Advanced Options**, on the **Encryption Key** tab select **Encrypt using customer-managed keys**.

**About Customer-Managed Encryption Key Rotation on Autonomous Database**

Describes how to rotate customer-managed encryption keys on Autonomous Database.

When you rotate the customer-managed master encryption key, Autonomous Database generates a new TDE master key and uses the new TDE master key to re-encrypt the tablespace encryption keys that encrypt and decrypt your data. This operation is fast and does not require database downtime. It does not change the tablespace keys and does not re-encrypt customer data.

**Note:**

Using the Oracle Cloud Infrastructure Console you can rotate an Oracle Cloud Infrastructure Vault master encryption key with the **Rotate Key** command. This is a separate action and does not result in a new master encryption key for your Autonomous Database. To rotate the master encryption key of your Autonomous Database, create a new master encryption key in Oracle Cloud Infrastructure Vault and follow the steps described below.

To rotate customer-managed encryption keys:

1. Create a new master encryption key in your Oracle Cloud Infrastructure Vault. If you already have multiple master encryption keys, then select a master encryption key that is different than the key you are using as your master encryption key for your Autonomous Database instance.

   See [Prerequisites to Use Customer-Managed Encryption Keys on Autonomous Database](#) for more information.

2. Rotate the master encryption key from the Oracle Cloud Infrastructure Console:

   See [Use Customer-Managed Encryption Keys on Autonomous Database](#) for more information.

**Prerequisites to Use Customer-Managed Encryption Keys on Autonomous Database**

Perform these prerequisite steps to use customer-managed keys on Autonomous Database:

1. Create an Oracle Cloud Infrastructure Vault.
a. Open the Oracle Cloud Infrastructure Console by clicking the menu next to Oracle Cloud.

b. From the Oracle Cloud Infrastructure left navigation menu click Identity and Security, and then click Vault.

c. Select an existing Vault or create a new Vault.

For more details, see the instructions for creating a vault, To create a new vault.

2. Create a Master Encryption Key in the Vault.

---

**Note:**

You must use these options when you create the key:

- **Key Shape: Algorithm:** AES (Symmetric key used for Encrypt and Decrypt)
- **Key Shape: Length:** 256 bits

For more information, see To create a new master encryption key and Overview of Key Management.

---

3. Create a dynamic group to make the master encryption key accessible to Autonomous Database.
32-5

Chapter 32
Prerequisites to Use Customer-Managed Encryption Keys on Autonomous Database

a. In the Oracle Cloud Infrastructure console click Identity and Security and click Dynamic Groups
b. Click Create Dynamic Group and enter a Name, a Description, and a rule.
c. Click Create.

**Create Dynamic Group for an existing database:**
You can specify that an Autonomous Database instance is part of the dynamic group. The dynamic group in the following example includes only the Autonomous Database whose OCID is specified in the resource.id parameter:

\[ resource.id = '{your_Autonomous_Database_instance_OCID}' \]

**Create a Dynamic Group for a database that has not been provisioned yet:**
When you are creating the dynamic group before you provision or clone an Autonomous Database instance, the OCID for the new database is not yet available. For this case, create a dynamic group that specifies the resources in a given compartment:

\[ resource.compartment.id = '{your_Compartment_OCID}' \]

See Perform Prerequisites to Use Resource Principal with Autonomous Database for more information.

4. Write policy statements for the dynamic group to enable access to Oracle Cloud Infrastructure resources (vaults and keys):
   a. In the Oracle Cloud Infrastructure console click Identity and Security and click Policies.
   b. To write policies for a dynamic group, click Create Policy, and enter a Name and a Description.
   c. Use the Policy Builder to create a policy.
      For example the following allows the members of the dynamic group DGKeyCustomer1 to access the vaults and keys in the compartment named training:

      Allow dynamic-group DGKeyCustomer1 to use vaults in compartment training
      Allow dynamic-group DGKeyCustomer1 to use keys in compartment training

      This sample policy applies for a single compartment. You can specify that a policy applies for your tenancy, a compartment, a resource, or a group of resources.
   d. Click Create.

See Managing Policies for more information on policies.
Use Customer-Managed Encryption Keys on Autonomous Database

Shows the steps to select customer-managed master encryption keys on Autonomous Database. If you are using customer-managed master encryption keys, follow these steps to rotate the master keys.

Caution:
The customer-managed encryption key is stored in Oracle Cloud Infrastructure Vault, external to the database host. If the customer-managed encryption key is disabled or deleted, the database will be inaccessible.

On Autonomous Database you can choose customer-managed keys as follows:

- While provisioning, under Advanced Options, on the Encryption Key tab select Encrypt using customer-managed keys.
- While cloning, under Advanced Options, on the Encryption Key tab select Encrypt using customer-managed keys.
- From an existing database, follow the steps in this section.

Follow these steps if your Autonomous Database is using Oracle-managed keys and you want to switch to customer-managed encryption keys, or if you are using customer-managed encryption keys and you want to rotate the master key.

1. Perform the required customer-managed encryption key prerequisite steps as necessary. See Prerequisites to Use Customer-Managed Encryption Keys on Autonomous Database for more information.
2. On the Details page, from the More Actions drop-down list, select Manage Encryption Key.
   If you are already using customer-managed keys and you want to rotate the TDE keys, follow these steps and select a different key (select a key that is different from the currently selected master encryption key).
4. Select a Vault.
   Click Change Compartment to select a vault in a different compartment.
5. Select a Master encryption key.
   Click Change Compartment to select a master encryption key in a different compartment.
6. Click **Save Changes**.

The **Lifecycle State** changes to **Updating**. When the request completes, the **Lifecycle State** shows **Available**.

After the request completes, on the Oracle Cloud Infrastructure Console, the key information shows on the Autonomous Database Information page under the heading **Encryption**. This area shows the **Encryption Key** field with a link to the Master Encryption Key and the **Encryption Key OCID** field with the Master Encryption Key OCID.

See **Notes for Using Customer-Managed Keys with Autonomous Database** for more information.

### Switch to Oracle-Managed Encryption Keys on Autonomous Database

Shows the steps to switch to Oracle-managed master encryption keys on Autonomous Database if you are using customer-managed encryption keys.

If your Autonomous Database is using customer-managed keys and you want to switch to Oracle-managed keys:

1. On the **Details** page, from the **More Actions** drop-down list, select **Manage Encryption Key**.
2. On the **Manage Encryption Key** page, select **Encrypt using Oracle-managed keys**.
3. Click **Save Changes**.

The **Lifecycle State** changes to **Updating**. When the request completes, the **Lifecycle State** shows **Available**.
Chapter 32

View History for Customer-Managed Encryption Keys on Autonomous Database

View History for Customer-Managed Encryption Keys on
Autonomous Database
You can view the key history from the Oracle Cloud Infrastructure Console or by
selecting from the V$ENCRYPTION_KEYS view.
To view the key history from the Oracle Cloud Infrastructure Console:
1.

On the Autonomous Database Details page, under Resources, click Key History.

2.

This shows the Key History.
For example:

To view the key history by selecting from the V$ENCRYPTION_KEYS view:
1.

Connect to the Autonomous Database as the ADMIN user.

2.

As the ADMIN user, issue the following command:
SELECT activation_time, tag FROM V$ENCRYPTION_KEYS;
For example:
SELECT activation_time, tag FROM V$ENCRYPTION_KEYS;
ACTIVATION_TIME
TAG
----------------------------------------------2020-02-19T17:21:57.821Z
{"credential_name":"OCI$RESOURCE_PRINCIPAL",
"oci_management_url":"https://aqmmanagement.kms.ca-toronto-1.oraclecloud.com/20180608/keys/",
"master_key_id":"ocid1.key.oc1.catoronto-1","username":"\"ADMIN\"",
"vault_id":"ocid1.vault.oc1.catoronto-1"}

See Notes for Using Customer-Managed Keys with Autonomous Database for more
information.

32-8


Notes for Using Customer-Managed Keys with Autonomous Database

Topics

• Caution for Customer-Managed Encryption Keys when Oracle Cloud Infrastructure Vault is Unreachable
• Caution for Using Customer-Managed Encryption Keys When the Master Key is Disabled or Deleted
• Caution for Using Customer-Managed Encryption Keys History and Backups
• Using Customer-Managed Keys with Autonomous Data Guard
• Using Customer-Managed Encryption Keys with Cloning

Caution for Customer-Managed Encryption Keys when Oracle Cloud Infrastructure Vault is Unreachable

After you switch to customer-managed keys, some database operations might be affected when Oracle Cloud Infrastructure Vault is unreachable, as follows:

If the database is unable to access Oracle Cloud Infrastructure Vault for some reason, such as a network outage, then Autonomous Database handles the outage as follows:

• There is a 2-hour grace period where the database remains up and running.
• If Oracle Cloud Infrastructure Vault is not reachable at the end of the 2-hour grace period, the database Lifecycle State is set to Inaccessible. In this state existing connections are dropped and new connections are not allowed.
• If Autonomous Data Guard is enabled, during or after the 2-hour grace period you can manually try to perform a failover operation. Autonomous Data Guard automatic failover is not triggered when you are using customer-managed encryption keys and the Oracle Cloud Infrastructure Vault is unreachable.
• If Autonomous Database is stopped, then you cannot start the database when the Oracle Cloud Infrastructure Vault is unreachable.

For this case, the work request shown when you click Work Requests on the Oracle Cloud Infrastructure console under Resources shows:

The Vault service is not accessible.
Your Autonomous Database could not be started. Please contact Oracle Support.

Caution for Using Customer-Managed Encryption Keys When the Master Key is Disabled or Deleted

After you switch to customer-managed keys, some database operations might be affected if the Oracle Cloud Infrastructure Vault key is disabled or deleted.
For disable/delete key operations where the Oracle Cloud Infrastructure Vault Master Encryption Key State is any of the following:

- DISABLING
- DISABLED
- DELETING
- DELETED
- SCHEDULING_DELETION
- PENDING_DELETION

The database becomes inaccessible after the Oracle Cloud Infrastructure Vault key lifecycleState goes into one of these states. When the Oracle Cloud Infrastructure Vault key is in any of these states, Autonomous Database drops existing connections and does not allow new connections.

If you disable or delete the Oracle Cloud Infrastructure Vault key used by your Autonomous Database while Autonomous Data Guard is enabled, Autonomous Data Guard will not perform an automatic failover.

If you enable a disabled key, the database automatically goes into the Available state.

If you delete the master key you can check the key history in the Oracle Cloud Infrastructure Console to find the keys that were used for the database. The history shows you which Oracle Cloud Infrastructure Vault key OCIDs were used, along with an activation timestamp. If one of the older keys from the history list is still available in the Vault, then you can restore the database to the time when the database was using that key, or you can clone from a backup to create a new database with that timestamp.

See View History for Customer-Managed Encryption Keys on Autonomous Database for more information.

Caution for Using Customer-Managed Encryption Keys History and Backups

After you switch to customer-managed keys, some database operations might be affected if the master key is rotated, disabled, or deleted and you do not have a valid key to restore your data from a previously saved backup or from a clone.

- It is recommended that you create a new Vault key that hasn’t been used for rotation in the last 60 days and use that for key rotation. This makes sure that you can go back to a backup if the current Vault key is deleted or disabled.

- When you perform multiple encryption key rotations within 60 days, it is recommended that you either use Oracle Cloud Infrastructure Vault to create a new key for each master encryption key rotation operation or specify a vault key OCID that has not been used in the last 60 days. This helps to save at least one vault key that you can use to recover your data from a backup, in the case where a customer-managed master encryption key is disabled or deleted.

See View History for Customer-Managed Encryption Keys on Autonomous Database for more information.
Using Customer-Managed Keys with Autonomous Data Guard

Autonomous Data Guard is fully supported when using customer-managed keys with a local standby database. Customer-managed keys are not supported when Autonomous Data Guard is enabled with a cross-region standby.

- If you are using Autonomous Data Guard with a local standby database and you do not enable an Autonomous Data Guard remote standby, when you select customer-managed keys the primary database and the local standby database use the same key. In the event of a switchover or failover to the local standby, you can continue to use the same key.
- If you disable or delete the Oracle Cloud Infrastructure Vault key used by your Autonomous Database while Autonomous Data Guard is enabled, Autonomous Data Guard does not perform an automatic failover.
- Switching to customer-managed keys is not allowed when Autonomous Data Guard is enabled with a remote standby.
- Enabling Autonomous Data Guard with a remote standby is not allowed if the Autonomous Database is using customer-managed keys.

Using Customer-Managed Encryption Keys with Cloning

- Autonomous Database does not support using customer-managed encryption keys with a refreshable clone. You cannot create a refreshable clone from a source database that uses customer-managed encryption keys. Additionally, you cannot switch to a customer-managed encryption key on a source database that has one or more refreshable clones.
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 and describes the secure client connection options. Also describes how to enable support for TLS connections (require mutual TLS only or allow both mutual TLS and TLS authentication).

Topics

• Configuring Network Access with Access Control Rules (ACLs)
• Configuring Network Access with Private Endpoints
• Update Network Options to Allow TLS or Require Only Mutual TLS (mTLS) Authentication on Autonomous Database

Configuring Network Access with Access Control Rules (ACLs)

Specifying an access control list blocks all IP addresses that are not in the ACL list from accessing the database. After you specify an access control list, the Autonomous Database only accepts connections from addresses on the access control list and the database rejects all other client connections.

Topics

• Configure Access Control Lists When You Provision or Clone an Instance
• Configure Access Control Lists for an Existing Autonomous Database Instance
• Change from Private to Public Endpoints with Autonomous Database
• Access Control List Restrictions and Notes

Configure Access Control Lists When You Provision or Clone an Instance

When you provision or clone Autonomous Database with the Secure access from allowed IPs and VCNs only option, you can restrict network access by defining an Access Control List (ACL).

See Provision Autonomous Database for information on provisioning your Autonomous Database.

Configure one or more Access Control Rules (ACLs), as follows:

1. In the Choose network access area, select Secure access from allowed IPs and VCNs only.

   With Secure access from allowed IPs and VCNs only selected, the console shows the fields and options to specify ACLs:
2. In the Choose network access area, specify the access control rules by selecting an **IP notation type** and entering **Values** appropriate for the type you select:

- **IP Address:**
  In **Values** field enter values for the **IP Address**. An IP address specified in a network ACL entry is the public IP address of the client that is visible on the public internet that you want to grant access. For example, for an Oracle Cloud Infrastructure VM, this is the IP address shown in the **Public IP** field on the Oracle Cloud Infrastructure console for that VM.

  
  [Note: Optionally click Add My IP Address to add your current IP address to the ACL entry.]

- **CIDR Block:**
  In **Values** field enter values for the **CIDR Block**. The CIDR block specified is the public CIDR block of the clients that are visible on the public internet that you want to grant access.

- **Virtual Cloud Network:**
  Use this option to specify the VCN for use with an Oracle Cloud Infrastructure Service Gateway. See **Access to Oracle Services: Service Gateway** for more information.

  - In **Virtual Cloud Network** field select the VCN that you want to grant access from. If you do not have the privileges to see the VCNs in your tenancy this list is empty. In this case use the selection **Virtual Cloud Network (OCID)** to specify the OCID of the VCN.

  - Optionally, in the **IP Addresses or CIDRs** field enter private IP addresses or private CIDR blocks as a comma separated list to whitelist specific clients in the VCN.

- **Virtual Cloud Network (OCID):**
Use this option to specify the VCN (OCID) for use with an Oracle Cloud Infrastructure Service Gateway. See Access to Oracle Services: Service Gateway for more information.

– In the Values field enter the OCID of the VCN you want to grant access from.
– Optionally, in the IP Addresses or CIDRs field enter private IP addresses or private CIDR blocks as a comma separated list to whitelist specific clients in the VCN.

If you want to specify multiple IP addresses or CIDR ranges within the same VCN, do not create multiple ACL entries. Use one ACL entry with the values for the multiple IP addresses or CIDR ranges separated by commas.

3. Click + Access Control Rule to add a new value to the access control list.

4. Click x to remove an entry.
   You can also clear the value in the IP Addresses or CIDR Blocks field to remove an entry.

5. Require mutual TLS (mTLS) authentication.
   After you enter an IP notation type and a value, you have the option to select this option. The options are:
   • When Require mutual TLS (mTLS) authentication is selected, only mTLS connections are allowed (TLS authentication is not allowed).
   • When Require mutual TLS (mTLS) authentication is deselected, TLS and mTLS connections are allowed. This is the default configuration.

   See Update Network Options to Allow TLS or Require Only Mutual TLS (mTLS) Authentication on Autonomous Database for more information.

6. Complete the remaining provisioning or cloning steps, as specified in Provision Autonomous Database, Clone an Autonomous Database Instance, or Clone Autonomous Database from a Backup.

After provisioning completes, you can update public endpoint ACLs or you can change the Autonomous Database configuration to use a private endpoint.

See Configure Access Control Lists for an Existing Autonomous Database Instance for information on updating ACLs.

See Change from Public to Private Endpoints with Autonomous Database for information on changing to a private endpoint.

Configure Access Control Lists for an Existing Autonomous Database Instance

You can control and restrict access to your Autonomous Database by specifying network access control lists (ACLs). On an existing Autonomous Database instance with a public endpoint you can add, change, or remove ACLs.

Configure ACLs, or add, remove, or update existing ACLs for an Autonomous Database instance as follows:

1. On the Details page, from the More Actions drop-down list, select Update Network Access.
   This shows the Update Network Access dialog.
2. In the dialog, under **Access Type**, select **Secure access from allowed IPs and VCNs only** and specify the access control rules by selecting an **IP notation type** and values:

- **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.

  ![Add My IP Address](image.png)

  Optionally click **Add My IP Address** to add your current IP address to the ACL entry.

- **CIDR Block:**
  In **Values** field enter values for the **CIDR Block**. The CIDR block specified is the public CIDR block of the clients that are visible on the public internet that you want to grant access.

- **Virtual Cloud Network:**
  Use this option to specify the VCN for use with an Oracle Cloud Infrastructure Service Gateway. See [Access to Oracle Services: Service Gateway](#) for more information.

    - In **Virtual Cloud Network** field select the VCN that you want to grant access from. If you do not have the privileges to see the VCNs in your tenancy this list is empty. In this case use the selection **Virtual Cloud Network (OCID)** to specify the OCID of the VCN.

    - Optionally, in the **IP Addresses or CIDRs** field enter private IP addresses or private CIDR blocks as a comma separated list to allow specific clients in the VCN.

- **Virtual Cloud Network (OCID):**
Use this option to specify the VCN for use with an Oracle Cloud Infrastructure Service Gateway. See Access to Oracle Services: Service Gateway for more information.

– In the **Values** field enter the OCID of the VCN you want to grant access from.
– Optionally, in the **IP Addresses or CIDRs** field enter private IP addresses or private CIDR blocks as a comma separated list to allow specific clients in the VCN.

If you want to specify multiple IP addresses or CIDR ranges within the same VCN, do not create multiple ACL entries. Use one ACL entry with the values for the multiple IP addresses or CIDR ranges separated by commas.

3. Click **+ Access Control Rule** to add a new value to the access control list.

4. Click x to remove an entry.

   You can also clear the value in the **IP Addresses** or **CIDR Blocks** field to remove an entry.

5. Click **Update.**

   If the Lifecycle State is **Available** when you click **Update** the Lifecycle State changes to **Updating** until the ACL is set. The database is still up and accessible, there is no downtime. When the update is complete the Lifecycle State returns to **Available** and the network ACLs from the access control list are in effect.

### Change from Private to Public Endpoints with Autonomous Database

If your Autonomous Database instance is configured to use a private endpoint you can change the configuration to use a public endpoint.

There are several prerequisites to change an instance from a private to a public endpoint, as follows:

- The Autonomous Database instance must be in the Available state (Lifecycle State: Available).
- Before changing the network configuration from a private endpoint to a public endpoint, you must change the configuration to not allow TLS connections. This closes any existing TLS connections. See Update your Autonomous Database Instance to Require mTLS and Disallow TLS Authentication for more information.

To specify a public endpoint for your Autonomous Database do the following:

1. On the **Details** page, from the **More Actions** drop-down list, select **Update Network Access**.

2. In the **Update Network Access** dialog, select one of **Secure access from everywhere** or **Secure access from allowed IPs and VCNs only**.

   For example, if you select **Secure access from allowed IPs and VCNs only** the dialog shows fields to configure access control rules:
3. In the dialog, under Configure access control rules specify rules by selecting an IP notation type and values:

- **IP Address**: In Values field enter values for the IP Address. An IP address specified in a network ACL entry is the public IP address of the client that is visible on the public internet that you want to grant access. For example, for an Oracle Cloud Infrastructure VM, this is the IP address shown in the Public IP field on the Oracle Cloud Infrastructure console for that VM.

  ![Image of IP Address field](image)

  Note:
  Optionally click **Add My IP Address** to add your current IP address to the ACL entry.

- **CIDR Block**: In Values field enter values for the CIDR Block. The CIDR block specified is the public CIDR block of the clients that are visible on the public internet that you want to grant access.

- **Virtual Cloud Network**: 
  - In Virtual Cloud Network field select the VCN that you want to grant access from. If you do not have the privileges to see the VCNs in your tenancy this list is empty. In this case use the selection **Virtual Cloud Network (OCID)** to specify the OCID of the VCN.
  - Optionally, in the IP Addresses or CIDRs field enter private IP addresses or private CIDR blocks as a comma separated list to allow specific clients in the VCN.

- **Virtual Cloud Network (OCID)**: 
  - In the Values field enter the OCID of the VCN you want to grant access from.
– Optionally, in the **IP Addresses or CIDRs** field enter private IP addresses or private CIDR blocks as a comma separated list to allow specific clients in the VCN.

If you want to specify multiple IP addresses or CIDR ranges within the same VCN, do not create multiple ACL entries. Use one ACL entry with the values for the multiple IP addresses or CIDR ranges separated by commas.

4. Click **Access Control Rule** to add a new value to the access control list.
5. Click **x** to remove an entry.

You can also clear the value in the **IP Addresses** or **CIDR Blocks** field to remove an entry.

6. Click **Update**.
7. In the **Confirm** dialog, type the Autonomous Database name to confirm the change.
8. In the **Confirm** dialog, click **Update**.

The Lifecycle State changes to **Updating** until the operation completes.

Notes for changing from private endpoint to public endpoint network access:

- After updating the network access type all database users must obtain a new wallet and use the new wallet to access the database. See **Download Client Credentials (Wallets)** for more information.
- After the update completes, you can change or define new access control rules ACLs for the public endpoint. See **Configure Access Control Lists for an Existing Autonomous Database Instance** for more information.
- The URL for Database Actions and for the Database Tools are different when a database uses a private endpoint compared to using a public endpoint. Click **Database Actions** on the Oracle Cloud Infrastructure Console to find the updated Database Actions URL and in **Database Actions** click the appropriate cards to find the updated Database Tools URLs, after changing from a private endpoint to a public endpoint.

Access Control List Restrictions and Notes

Describes restrictions and notes for access control rules on Autonomous Database.

- If you want to only allow connections coming through a service gateway you need to use the IP address of the service gateway in your ACL definition. To do this you need to add an ACL definition with the CIDR source type with the value 240.0.0.0/4. Note that this is not recommended, instead of this you can specify individual VCNs in your ACL definition for the VCNs you want to allow access from.

See **Access to Oracle Services: Service Gateway** for more information.

- When you restore a database the existing ACLs are not overwritten by the restore.

- The network ACLs apply to the database connections and Oracle Machine Learning notebooks. If an ACL is defined, if you try to login to Oracle Machine Learning Notebooks from a client whose IP is not specified on the ACL this shows the "login rejected based on access control list set by the administrator" error.

- The following Autonomous Database tools are subject to ACLs. You can use Virtual Cloud Network, Virtual Cloud Network (OCID), IP address, or CIDR block ACLs to control access to these tools:
  - **Database Actions**
– Oracle APEX
– Oracle Graph Studio
– Oracle Machine Learning Notebooks
– Oracle REST Data Services

• If you have a private subnet in your VCN that is configured to access the public internet through a NAT Gateway, you need to enter the public IP address of the NAT Gateway in your ACL definition. Clients in the private subnet do not have public IP addresses. See NAT Gateway for more information.

• If you are using ACLs and TLS connections are allowed, you must change your network configuration to not allow TLS connections before removing all ACLs. See Update your Autonomous Database Instance to Require mTLS and Disallow TLS Authentication for more information.

Configuring Network Access with Private Endpoints

You can specify that Autonomous Database uses a private endpoint inside your Virtual Cloud Network (VCN) in your tenancy. You can configure a private endpoint during provisioning or cloning your Autonomous Database, or you can switch to using a private endpoint in an existing database that uses a public endpoint. This allows you to keep all traffic to and from your database off of the public internet.

Specifying the virtual cloud network configuration allows traffic only from the virtual cloud network you specify and blocks access to the database from all public IPs or VCNs. This allows you to define security rules with Security Lists or at the Network Security Group (NSG) level to specify ingress/egress for your Autonomous Database instance. Using a private endpoint and defining Security Lists or NSGs allows you to control traffic to and from your Autonomous Database instance.

Topics

• Configure Private Endpoints
• Enhanced Security for Outbound Connections with Private Endpoints
• Private Endpoints Notes
• Private Endpoints Configuration Examples on Autonomous Database

Configure Private Endpoints

You can specify that Autonomous Database uses a private endpoint and configure a Virtual Cloud Network (VCN) in your tenancy to use with the private endpoint.

Perform the following prerequisite steps before configuring a private endpoint:

• Set required policies for the resources you are working with. See Prerequisite: IAM Policies Required to Manage Private Endpoints for more information.

• Create a VCN within the region that will contain your Autonomous Database. See VCNs and Subnets for more information.

• Configure a subnet within your VCN configured with default DHCP options. See DNS in Your Virtual Cloud Network for more information.

(Optional) Perform the following optional step before configuring a private endpoint:
- Specify a Network Security Group (NSG) within your VCN. The NSG specifies rules for connections to your Autonomous Database. See Network Security Groups for more information.

You can configure the private endpoint for an existing Autonomous Database instance or when you provision or clone a new instance:

- Change from Public to Private Endpoints with Autonomous Database
- Configure Private Endpoints When You Provision or Clone an Instance

### Prerequisite: IAM Policies Required to Manage Private Endpoints

Autonomous Database relies on the IAM (Identity and Access Management) service to authenticate and authorize cloud users to perform operations that use any of the Oracle Cloud Infrastructure interfaces (the Console, REST API, CLI, SDK, or others).

The IAM service uses **groups**, **compartments** and **policies** to control which cloud users can access which resources. In particular, a policy defines what kind of access a group of users has to a particular kind of resource in a particular compartment. For more information, see Getting Started with Policies.

In addition to the policies required to provision and manage an Autonomous Database, some network policies are needed to use private endpoints. The following table lists the IAM policies required for a cloud user to add a private endpoint.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Required IAM Policies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure a private endpoint</td>
<td>use vcns for the compartment which the VCN is in</td>
</tr>
<tr>
<td></td>
<td>use subnets for the compartment which the VCN is in</td>
</tr>
<tr>
<td></td>
<td>use network-security-groups for the compartment which the network security group is in</td>
</tr>
<tr>
<td></td>
<td>manage private-ips for the compartment which the VCN is in</td>
</tr>
<tr>
<td></td>
<td>manage vnics for the compartment which the VCN is in</td>
</tr>
<tr>
<td></td>
<td>manage vnics for the compartment which the database is provisioned or is to be provisioned in</td>
</tr>
</tbody>
</table>

See Common Policies for more information.

---

**Note:**

The listed policies are the minimum requirements to add a private endpoint. You can also use a policy rule that is broader. For example, if you set the policy rule:

```
Allow group MyGroupName to manage virtual-network-family in tenancy
```

This rule also works because it is a superset that contains all the required policies.
Configure Private Endpoints When You Provision or Clone an Instance

You can configure a private endpoint when you provision or clone an Autonomous Database instance.

These steps assume you are provisioning or cloning an instance and you have completed the prerequisite steps, and you are at the Choose network access step of the provisioning or cloning steps:

1. Select **Private endpoint access only**.
   
   This expands the Virtual cloud network private access configuration area.
Note:

If you select **Private endpoint access only**, this only allows connections from the specified private network (VCN), from peered VCNs, and from on-prem networks connected to your VCN. Thus, you can configure an Autonomous Database instance on a private endpoint to allow connections from on-prem networks. See Example: Connecting from Your Data Center to Autonomous Database for an example.

If you want to allow connections from public IP addresses, then you need to select either **Secure access from everywhere** or **Secure access from allowed IPs and VCNs only** when you provision or clone your Autonomous Database.

2. Select a **Virtual cloud network** in your compartment or if the VCN is in a different compartment click **Change Compartment** and select the compartment that contains the VCN and then select a virtual cloud network.

See VCNs and Subnets for more information.

3. Select the **Subnet** in your compartment to attach the Autonomous Database to or if the Subnet is in a different compartment click **Change Compartment** and select the compartment that contains the Subnet and then select a subnet.

See VCNs and Subnets for more information.

4. (Optional) Enter a **Hostname prefix**.

This specifies a hostname prefix for the Autonomous Database and associates a DNS name with the database instance, in the following form:

```
hostname_prefix.adb.region.oraclecloud.com
```

If you do not specify a hostname prefix, a system generated hostname prefix is supplied.

5. (Optional) Add **Network security groups (NSGs)**.

Optionally, to allow connections to the Autonomous Database instance define security rules in an NSG; this creates a virtual firewall for your Autonomous Database.

a. Select a Network Security Group in your compartment to attach the Autonomous Database to, or if the Network Security Group is in a different compartment, click **Change Compartment** and select a different compartment and then select a Network Security Group in that compartment.

b. Click **+ Another Network Security Group** to add another Network Security Group.

c. Click **x** to remove a Network Security Group entry.

For the NSG you select for the private endpoint define a security rule as follows:

- For mutual TLS (mTLS) authentication, add a stateful ingress rule with the source set to the address range you want to allow to connect to your database, the IP Protocol set to TCP, and the Destination Port Range set to 1522. See About Mutual TLS (mTLS) Authentication for more information.

- For TLS authentication, add a stateful ingress rule with the source set to the address range you want to allow to connect to your database, the IP Protocol set to TCP, and the Destination Port Range set to 1521. See About TLS Authentication for more information.
• To use Oracle APEX, Database Actions, and Oracle REST Data Services, add port 443 to the NSG rule.

![Note:]

Incoming and outgoing connections are limited by the combination of ingress and egress rules defined in NSGs and the Security Lists defined with the VCN. When there are no NSGs, ingress and egress rules defined in the Security Lists for the VCN still apply. See Security Lists for more information on working with Security Lists.

See Private Endpoints Configuration Examples on Autonomous Database for examples.
See Network Security Groups for more information.

6. Require mutual TLS (mTLS) authentication.

The Require mutual TLS (mTLS) authentication options are:
• When Require mutual TLS (mTLS) authentication is deselected, TLS and mTLS connections are allowed. This is the default configuration.
• When Require mutual TLS (mTLS) authentication is selected, only mTLS connections are allowed (TLS authentication is not allowed).

See Update Network Options to Allow TLS or Require Only Mutual TLS (mTLS) Authentication on Autonomous Database for more information.

7. Complete the remaining provisioning or cloning steps, as specified in Provision Autonomous Database, Clone an Autonomous Database Instance, or Clone Autonomous Database from a Backup.
See Private Endpoints Notes for more information.

Change from Public to Private Endpoints with Autonomous Database

If your Autonomous Database instance is configured to use a public endpoint you can change the configuration to a private endpoint.

1. On the Details page, from the More Actions drop-down list, select Update Network Access.

   To change an instance from a public to a private endpoint, the Autonomous Database instance must be in the Available state (Lifecycle State: Available).

2. In the Update Network Access dialog, select Private endpoint access only. This expands the Virtual cloud network private access configuration area.
Chapter 33
Configuring Network Access with Private Endpoints

Update Network Access

Choose network access

Access Type
- Secure access from everywhere
  Allow users with database credentials to access the database from the internet.
- Secure access from allowed IPs and VCNs only
  Restrict access to specified IP addresses and VCNs.
- Private endpoint access only
  Restrict access to a private endpoint within an OCI VCN.

Virtual cloud network in sic-dbaas (root) (Change Compartment)
Select a Virtual Cloud Network

Subnet in sic-dbaas (root) (Change Compartment)
Select a Virtual Cloud Network

Host name prefix (Optional)

The name can contain only letters and numbers and a maximum of 63 characters.

Network security groups (NSGs)

Network Security Groups in sic-dbaas (root) (Change Compartment)
Select a Virtual cloud network first

+ Another Network Security Group

Update  Cancel
3. 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.

4. 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.

5. **(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:

`hostname_prefix.adb.region.oraclecloud.com`

If you do not specify a hostname prefix, a system generated hostname prefix is supplied.

6. **(Optional)** Add **Network security groups (NSGs)**.

Optionally, add one or more NSGs to allow connections to the Autonomous Database instance and define security rules for the NSGs; this creates a virtual firewall for your Autonomous Database.

   a. Select a Network Security Group in your compartment to attach the Autonomous Database to, or if the Network Security Group is in a different compartment, click **Change Compartment** and select a different compartment and then select a Network Security Group in that compartment.

   b. Click **+ Another Network Security Group** to add another Network Security Group.

   c. Click **x** to remove a Network Security Group entry.

For the NSG you select for the private endpoint define a security rule as follows:

- For mutual TLS authentication, add a stateful ingress rule with the source set to the address range you want to allow to connect to your database, the IP Protocol set to TCP, and the Destination Port Range set to 1522. See About Mutual TLS (mTLS) Authentication for more information.

- For TLS authentication, add a stateful ingress rule with the source set to the address range you want to allow to connect to your database, the IP Protocol set to TCP, and the Destination Port Range set to 1521. See About TLS Authentication for more information.
• To use Oracle APEX, Database Actions, and Oracle REST Data Services, add port 443 to the NSG rule.

Note:

Incoming and outgoing connections are limited by the combination of ingress and egress rules defined in NSGs and the Security Lists defined with the VCN. When there are no NSGs, ingress and egress rules defined in the Security Lists for the VCN still apply. See Security Lists for more information on working with Security Lists.

See Private Endpoints Configuration Examples on Autonomous Database for examples. See Network Security Groups for more information.

7. Click Update.
8. In the Confirm dialog, type the Autonomous Database name to confirm the change.
9. In the Confirm dialog, click Update.

The Lifecycle State changes to Updating until the operation completes.

Notes for changing from public to private network access:

• After updating the network access type all database users must obtain a new wallet and use the new wallet to access the database. See Download Client Credentials (Wallets) for more information.
• If you had ACLs defined for the public endpoint, the ACLs do not apply for the private endpoint.
• After you update the network access to use a private endpoint, the URL for the Database Tools is different compared to using a public endpoint. You can find the updated URLs on the console, after changing from a public endpoint to a private endpoint.

Enhanced Security for Outbound Connections with Private Endpoints

When you define a private endpoint for your Autonomous Database instance you can provide enhanced security by setting a database property to enforce that all outgoing connections to a target host are subject to and limited by the private endpoint's egress rules. You define egress rules in the Virtual Cloud Network (VCN) security list or in the Network Security Group (NSG) associated with the Autonomous Database instance private endpoint.

Before you set this database property configure your Autonomous Database instance to use a private endpoint. See Configure Private Endpoints for more information.

Set the ROUTE_OUTBOUND_CONNECTIONS database property to PRIVATE_ENDPOINT to specify that all outgoing connections are subject to the Autonomous Database instance private endpoint VCN's egress rules. With the value PRIVATE_ENDPOINT the database restricts outgoing connections to locations specified by the private endpoint's egress rules.
Note:

With `ROUTE_OUTBOUND_CONNECTIONS` not set to `PRIVATE_ENDPOINT`, all outgoing connections to the public internet pass through the Network Address Translation (NAT) Gateway of the service VCN. In this case, if the target host is on a public endpoint the outgoing connections are not subject to the Autonomous Database instance private endpoint VCN or NSG egress rules.

When you configure a private endpoint for your Autonomous Database instance and set `ROUTE_OUTBOUND_CONNECTIONS` to `PRIVATE_ENDPOINT`, this setting changes the handling of outbound connections for the following:

- Database links
- APEX_LDAP, APEX_MAIL, and APEX_WEB_SERVICE
- UTL_HTTP, UTL_SMTP, and UTL_TCP
- DBMS_LDAP

To set `ROUTE_OUTBOUND_CONNECTIONS`:

1. Connect to your database.
2. Set the database property `ROUTE_OUTBOUND_CONNECTIONS`.

   For example:

   ```sql
   ALTER DATABASE PROPERTY SET ROUTE_OUTBOUND_CONNECTIONS = 'PRIVATE_ENDPOINT';
   ```

Notes for setting `ROUTE_OUTBOUND_CONNECTIONS`:

- Use the following command to restore the default parameter value:

  ```sql
  ALTER DATABASE PROPERTY SET ROUTE_OUTBOUND_CONNECTIONS = '';
  ```

- Use the following command to query the current parameter value:

  ```sql
  SELECT * FROM DATABASE_PROPERTIES
  WHERE PROPERTY_NAME = 'ROUTE_OUTBOUND_CONNECTIONS';
  ```

If the property is not set the query does not return results.

- This property only applies for database links that you create after you set the property to the value `PRIVATE_ENDPOINT`. Thus, database links that you created prior to setting the property continue to use the NAT Gateway of the service VCN and are not subject to the Autonomous Database instance private endpoint's egress rules.

- Only set `ROUTE_OUTBOUND_CONNECTIONS` to the value `PRIVATE_ENDPOINT` when you are using Autonomous Database with a private endpoint.

- By default, when you are accessing other private endpoints, the connection is subject to your VCN's egress rules. Setting `ROUTE_OUTBOUND_CONNECTIONS` has no effect in this case. The `ROUTE_OUTBOUND_CONNECTIONS` property applies when you...
want outgoing connections to follow the private endpoint egress rules even when accessing public endpoints.

See NAT Gateway for more information on Network Address Translation (NAT) gateway.

Private Endpoints Notes

Describes restrictions and notes for private endpoints on Autonomous Database.

- After you update the network access to use a private endpoint, or after the provisioning or cloning completes where you configure a private endpoint, you can view the network configuration on the Autonomous Database Details page under the Network section.

  The Network section shows the following information for a private endpoint:
  - **Access Type**: Specifies the access type for the Autonomous Database configuration. Private endpoint configurations show the access type: Virtual Cloud Network.
  - **Virtual Cloud Network**: This includes a link for the VCN associated with the private endpoint.
  - **Subnet**: This includes a link for the subnet associated with the private endpoint.
  - **Private IP**: Shows the private IP for the private endpoint configuration.
  - **Network Security Groups**: This field includes links to the NSG(s) configured with the private endpoint.

- After provisioning or cloning completes, you can change the Autonomous Database configuration to use a public endpoint.

  See Change from Private to Public Endpoints with Autonomous Database for information on changing to a public endpoint.

- You can specify up to five NSGs to control access to your Autonomous Database.

- You can change the private endpoint Network Security Group (NSG) for the Autonomous Database.

  To change the NSG for a private endpoint, do the following:

  1. On the Autonomous Databases page select an Autonomous Database from the links under the Display Name column.


- You can connect your Oracle Analytics Cloud instance to your Autonomous Database that has a private endpoint using the Data Gateway like you do for an on-premises database. See Configure and Register Data Gateway for Data Visualization for more information.

- The following Autonomous Database tools are supported in databases configured with a private endpoint:
  - Database Actions
  - Oracle APEX
  - Oracle Graph Studio
  - Oracle Machine Learning Notebooks
  - Oracle REST Data Services
  - Oracle Database API for MongoDB
Additional configuration is required to access these Autonomous Database tools from on-premises environments. See Example: Connecting from Your Data Center to Autonomous Database to learn more.

Accessing Oracle APEX, Database Actions, Oracle Graph Studio, or Oracle REST Data Services using a private endpoint from on-premises environments without completing the additional private endpoint configuration shows the error:

```
404 Not Found
```

- After you update the network access to use a private endpoint, the URL for the Database Tools is different compared to using a public endpoint. You can find the updated URLs on the console, after changing from a public endpoint to a private endpoint.

- In addition to the default Oracle REST Data Services (ORDS) preconfigured with Autonomous Database, you can configure an alternative ORDS deployment that provides more configuration options and that can be used with private endpoints. See About Customer Managed Oracle REST Data Services on Autonomous Database to learn about an alternative ORDS deployment that can be used with private endpoints.

Private Endpoints Configuration Examples on Autonomous Database

Shows several Private Endpoint (VCN) configuration samples for Autonomous Database.

**Topics**

- Example: Connecting from Inside Oracle Cloud Infrastructure VCN
- Example: Connecting from Your Data Center to Autonomous Database

**Example: Connecting from Inside Oracle Cloud Infrastructure VCN**

Demonstrates an application running inside Oracle Cloud Infrastructure on a virtual machine (VM) in the same VCN which is configured with your Autonomous Database.
There is an Autonomous Database instance which has a private endpoint in the VCN named "Your VCN". The VCN includes two subnets: "SUBNET B" (CIDR 10.0.1.0/24) and "SUBNET A" (CIDR 10.0.2.0/24).

The Network Security Group (NSG) associated with the Autonomous Database instance is shown as "NSG 1 - Security Rules". This Network Security Group defines security rules that allow incoming and outgoing traffic to and from the Autonomous Database instance. Define a rule for the Autonomous Database instance as follows:

- For Mutual TLS authentication, add a stateful ingress rule to allow connections from the source to the Autonomous Database instance; the source is set to the address range you want to allow to connect to your database, IP Protocol is set to TCP, and the Destination Port Range is set to 1522.
- For TLS authentication, add a stateful ingress rule to allow connections from the source to the Autonomous Database instance; the source is set to the address range you want to allow to connect to your database, IP Protocol is set to TCP, and the Destination Port Range is set to 1521.
- To use Oracle APEX, Database Actions, and Oracle REST Data Services, add port 443 to the NSG rule.

The following figure shows a sample stateful security rule to control traffic for the Autonomous Database instance:

The application connecting to the Autonomous Database is running on a VM in SUBNET B. You also add a security rule to allow traffic to and from the VM (as shown, with label "NSG 2 Security Rules"). You can use a stateful security rule for the VM, so simply add a rule for egress to NSG 2 Security Rules (this allows access to the destination subnet A).
The following figure shows sample security rules that control traffic for the VM:

After you configure the security rules, your application can connect to the Autonomous Database instance using the client credentials wallet. See Download Client Credentials (Wallets) for more information.


Example: Connecting from Your Data Center to Autonomous Database

Demonstrates how to connect privately to an Autonomous Database from your on-premise data center. In this scenario, traffic never goes over the public internet.

To connect from your data center, you connect the on-premise network to the VCN with FastConnect and then set up a Dynamic Routing Gateway (DRG). To resolve the Autonomous Database private endpoint, a Fully Qualified Domain Name (FQDN), requires that you add an entry in your on-premise client's hosts file. For example, /etc/hosts entry -> 10.0.2.7 example.adb.ca-toronto-1.oraclecloud.com

To use Oracle APEX, Database Actions, and Oracle REST Data Services, add another entry with the same IP. For example:

/etc/hosts entry -> 10.0.2.7 example.adb.ca-toronto-1.oraclecloudapps.com

You find the private endpoint IP and the FQDN as follows:
• The Private IP is shown on the Oracle Cloud Infrastructure console Autonomous Database details page for the instance.

• The FQDN is shown in the tnsnames.ora file in the Autonomous Database client credential wallet.

Alternatively you can 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:

• For Mutual TLS authentication, create a stateful rule to allow ingress traffic from the data center. This is a stateful ingress rule with the source set to the address range you want to allow to connect to your database, protocol set to TCP, source port range set to CIDR range (172.16.0.0/16), and destination port set to 1522.

• For TLS authentication, create a stateful rule to allow ingress traffic from the data center. This is a stateful ingress rule with the source set to the address range you want to allow to connect to your database, protocol set to TCP, source port range set to CIDR range (172.16.0.0/16), and destination port set to 1521.

• To use Oracle APEX, Database Actions, and Oracle REST Data Services, add port 443 to the NSG rule.

The following figure shows the security rule that controls traffic for the Autonomous Database instance:

After you configure the security rule, your on-premise database application can connect to the Autonomous Database instance using the client credentials wallet. See Download Client Credentials (Wallets) for more information.

Update Network Options to Allow TLS or Require Only Mutual TLS (mTLS) Authentication on Autonomous Database

Describes how to update the secure client connection authentication options, Mutual TLS (mTLS) and TLS.

Topics

• Network Access Prerequisites for TLS Connections

• Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication

• Update your Autonomous Database Instance to Require mTLS and Disallow TLS Authentication
Network Access Prerequisites for TLS Connections

Describes the network access configuration prerequisites for TLS connections.

To allow an Autonomous Database instance to use TLS connections, either ACLs must be defined or a private endpoint must be configured:

• When an Autonomous Database instance is configured to operate over the public internet, one or more Access Control Lists (ACLs) must be defined before you use TLS authentication to connect to the database. To validate that ACLs are defined, in the Network area on the Autonomous Database Details page view the Access Control List field. This field shows Enabled when ACLs are defined and shows Disabled when ACLs are not defined.

See Configuring Network Access with Access Control Rules (ACLs) for more information.

• When an Autonomous Database instance is configured with a private endpoint you can use TLS authentication to connect to the database. To validate that a private endpoint is defined, in the Network area on the Autonomous Database Details page view the Access Type field. This field shows Virtual Cloud Network when a private endpoint is defined.


Note:

When an Autonomous Database instance is configured with the network access type: Allow secure access from everywhere, you can only use TLS connections to connect to the database if you specify ACLs to restrict access.

Update your Autonomous Database Instance to Allow both TLS and mTLS Authentication

If your Autonomous Database instance is configured to only allow mTLS connections, you can update the instance to allow both mTLS and TLS connections. When you update your configuration to allow both mTLS and TLS, you can use both authentication types at the same time and connections are no longer restricted to require mTLS authentication.

You can allow TLS connections when network access is configured as follows:

• With network access configured with ACLs defined.
• With network access configured with a private endpoint defined.

Note:

TLS connections are only supported in commercial regions. Government regions only allow mTLS connections to an Autonomous Database instance.
Perform the network access configuration prerequisites. See [Network Access Prerequisites for TLS Connections](#) for more information.

Perform the following steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click [Oracle Database](#) and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display Name** column.

To change the Autonomous Database instance to allow TLS authentication, do the following:

1. On the **Autonomous Database Details** page, under **Network**, click **Edit** in the **Mutual TLS (mTLS) Authentication** field.
   This shows the Edit Mutual TLS Authentication page.

2. To change the value to allow TLS authentication, deselect **Require mutual TLS (mTLS) authentication**.

3. Click **Update**.
   The Autonomous Database Lifecycle State changes to **Updating**.
   After some time, the **Lifecycle State** shows **Available** and the **Mutual TLS (mTLS) Authentication** field changes to show **Not Required**.

Depending on the type of client, TLS connections have the following support with Autonomous Database:

- If the client is connecting with JDBC Thin using TLS authentication, the client can connect without providing a wallet. See [Connect with JDBC Thin Driver](#) for more information.
- If the client is connecting with managed ODP.NET or ODP.NET Core versions 19.13 or 21.4 (or above) using TLS authentication, the client can connect without providing a wallet. See [Connect Microsoft .NET, Visual Studio Code, and Visual Studio without a Wallet](#) for more information.
- If the client is connecting with SQLNet and Oracle Call Interface (OCI), and for certain other connection types with TLS authentication, the clients must provide the CA certificate in a wallet. See [Prepare for Oracle Call Interface, ODBC, and JDBC OCI](#) for more information.
Connections Using TLS Authentication and Connect SQL*Plus without a Wallet for more information.

Update your Autonomous Database Instance to Require mTLS and Disallow TLS Authentication

If your Autonomous Database instance is configured to allow TLS connections, you can update the instance to require mTLS connections and disallow TLS connections.

**Note:**

When you update an Autonomous Database instance to require Mutual TLS (mTLS) connections, existing TLS connections are disconnected.

Perform the following steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the Display Name column.

To change the Autonomous Database instance to require mTLS authentication and to not allow TLS authentication, do the following:

1. On the Autonomous Database Details page, under Network, click Edit in the Mutual TLS (mTLS) Authentication field.
   
   This shows the Edit Mutual TLS Authentication page.

2. Select Require mutual TLS (mTLS) authentication.

![Edit Mutual TLS Authentication](image)
3. Click **Update**.

The Autonomous Database Lifecycle State changes to **Updating**.

After some time, the **Lifecycle State** shows **Available** and the **Mutual TLS (mTLS) Authentication** field changes to show **Required**.
Managing and Viewing Maintenance Windows, Patching Options, Time Zone Data Updates, Work Requests, Events and Notifications, and Customer Contacts

Oracle Cloud Infrastructure Events enable you to create automation based on state changes for your Autonomous Database. Oracle Cloud Infrastructure work requests allow you to monitor long-running operations such as cloning or backing up a database. Oracle Cloud Infrastructure Console provides information on Autonomous Database maintenance windows and patch level and provides a customer contacts area where you can view or manage the list of contacts to receive information on operational issues and service announcements. This chapter also provides information about time zone data file updates.

Topics

• About Work Requests
• View Patch and Maintenance Window Information, Set the Patch Level
• View and Manage Customer Contacts for Operational Issues and Announcements
• Use Autonomous Database Events
• Manage Time Zone File Version on Autonomous Database

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.

See **Work Requests Integration** for more information.

**View Patch and Maintenance Window Information, Set the Patch Level**

Autonomous Database uses predefined maintenance windows to automatically patch your database. You can view maintenance and patch information and see details for Autonomous Database maintenance history. When you provision your database you can select a patch level.

**Topics**

- About Scheduled Maintenance and Patching
- View Maintenance Event History
- View Patch Details
- Set the Patch Level
- View Maintenance Status and Timezone Version Notifications

**About Scheduled Maintenance and Patching**

All Autonomous Database instances are automatically assigned to a maintenance window and different instances can have different maintenance windows.

The Autonomous Database Details page shows the **Next Maintenance** field that includes the date and time for the upcoming maintenance window; the date is updated automatically when the next maintenance window is scheduled. The **View History** link provides details on past maintenance. The **Patch Level** field shows the patch level setting for the Autonomous Database instance.

**Note:**

Your database remains available during the maintenance window. Your existing database connections may get disconnected briefly; however, you can immediately reconnect and continue using your database.
If your database is in the stopped state during the maintenance window, the database changes from the patch are applied when you start your database.

You can use Oracle Cloud Infrastructure Events to be notified when maintenance begins and ends. See About Information Events on Autonomous Database for more information.

View Maintenance Event History

You can view Autonomous Database maintenance event history for details about past maintenance events, such as the title, state, start time, and stop time.

Perform the following prerequisite steps as necessary:

• Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
• From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
• On the Autonomous Databases page select an Autonomous Database from the links under the Display Name column.

To view maintenance history, do the following:

1. On the Autonomous Database Details page, under Maintenance, click View History.
2. The Oracle Cloud Infrastructure Console displays the Maintenance History page.
3. (Optional) Use the State selector to filter events by state.

For example, if you select Succeeded, the Maintenance History page shows only the maintenance events with the Succeeded state.

The Maintenance History page shows details for each maintenance event, including the following:

• **Title**: The name of the maintenance event.
• **Maintenance Type**: Planned or Unplanned.
• **Resource Type**: The type of the resource on which the maintenance event occurs: Database or Infrastructure.
• **State**: Succeeded, Failed, or In progress.
• **Start Time**: Maintenance start time.
• **End Time**: Maintenance end time.
Note:

Maintenance event history is available starting with maintenance events after February 2021.

View Patch Details

You can view Autonomous Database patch information, including a list of resolved issues and components.

To view patch information for a specific patch, do the following:

1. Select the Autonomous Database patch that you want to view. The Maintenance History page on the Oracle Cloud Infrastructure Console shows the list of patches under Title.
2. For the selected patch, query the DBA_CLOUD_PATCH_INFO view.
   For example, for patch ADBS-21.7.1.1, use the following query:
   ```sql
   SELECT * FROM DBA_CLOUD_PATCH_INFO WHERE PATCH_VERSION = 'ADBS-21.7.1.1';
   ```
3. For an issue of interest, query the view to obtain details for the issue:
   ```sql
   SELECT * FROM DBA_CLOUD_PATCH_INFO WHERE PATCH_VERSION = 'ADBS-21.7.1.1' and BUG_NUM = bug_number;
   ```

To view patch information for all available patches:

```sql
SELECT * FROM DBA_CLOUD_PATCH_INFO;
```

Notes for viewing patch information:

- The view DBA_CLOUD_PATCH_INFO is available to the ADMIN user.
- Patch information and details on resolved issues is available from ADBS-21.7.1.1 onwards (starting in July 2021).
- The view DBA_CLOUD_PATCH_INFO has the following columns:

  BUG_NUM, BUG_TITLE, COMPONENT_NAME, PATCH_VERSION

Set the Patch Level

When you provision or clone an Autonomous Database instance you can select a patch level to apply to upcoming patches. There are two patch level options: Regular and Early.

When you select patch level Early, patches are applied for the Autonomous Database instance one week before the Regular scheduled patch. The Next Maintenance field in the Oracle Cloud Infrastructure Console reflects a maintenance window date and time based on the patch level.
The default patch level for provisioning an Autonomous Database instance is Regular. The default patch level for cloning is the patch level specified for the source database. Provisioning or cloning an instance and setting the patch level to Early allows you to use and to test upcoming patches before they are applied to all systems.

**Note:**

When cloning a source database with Early patch level, you can only choose Early patch level for your clone.

To set the patch level, do the following:

- Set the patch level when you provision or clone an instance.

  When you provision a new instance, follow the provisioning instructions and select the patch level, either Regular or Early. See Provision Autonomous Database for more information.

  When you clone an instance, follow the cloning instructions and select a patch level, either Regular or Early. See Clone an Autonomous Database Instance for more information.

To change the patch level, do the following:

1. You cannot change the patch level for an existing Autonomous Database instance. The option to set the patch level is only available when you provision or clone an Autonomous Database instance.

2. Change the patch level by cloning a new instance and selecting a different patch level for the cloned database. Cloning a source database with the patch level Regular to Early is allowed. Cloning a source database with the patch level Early to Regular is not allowed. See Clone an Autonomous Database Instance for more information.

Reporting Patch Issues to Oracle Support

Oracle Support provides the same handling for regular or early patch level Autonomous Databases. If you are using an Autonomous Database instance and the patch level is Early, Oracle Support considers issues you report with high priority and after validating an issue, determines if the patch should be applied or withheld from the upcoming Regular patch.

If you have an issue to report, file a service request at Oracle Cloud Support or contact your support representative.

Notes for patching level:

- The option to set the patch level is not available in every region. In some regions all Autonomous Database instances are provisioned or cloned at the Regular patch level.

- Autonomous Data Guard is only available for instances with patch level Regular. When you configure an Autonomous Database instance with patch level Early, you cannot enable Autonomous Data Guard.

- Always Free Autonomous Database instances do not provide the Early patch level option.

- You cannot set the patch level when you create a refreshable clone. A refreshable clone has the same patch level as the source database.
View Maintenance Status and Timezone Version Notifications

The **DB_NOTIFICATIONS** view stores information about maintenance status notifications and timezone version upgrade notifications for your Autonomous Database instance.

**To show notification information:**

1. Connect to your Autonomous Database instance.
2. Use the following query to view maintenance (patching) information.

   ```sql
   SELECT * FROM DB_NOTIFICATIONS WHERE TYPE = 'MAINTENANCE';
   ```

3. Use the following query to view timezone version information. The timezone version notification is only shown if the timezone version is not the latest. The description provides details on how to upgrade to the latest version.

   ```sql
   SELECT DESCRIPTION FROM DB_NOTIFICATIONS WHERE TYPE = 'TIMEZONE_VERSION';
   ```

The following provides details about the **DESCRIPTION** field values.

- **Maintenance run has ended**: Specifies maintenance has completed. The **MAINTENANCE_STATUS** shows the value **COMPLETED** with the start and end timestamps for the completed maintenance in **ACTUAL_START_DATE** and **ACTUAL_END_DATE**.

- **Maintenance run is scheduled for the instance**: Specifies a new maintenance has been scheduled. The **MAINTENANCE_STATUS** shows the value **SCHEDULED** with the expected start and end timestamps for the scheduled maintenance in **EXPECTED_START_DATE** and **EXPECTED_END_DATE**.

- **Maintenance run has begun**: Specifies the maintenance is in progress and provides the start timestamp for the active maintenance. The **MAINTENANCE_STATUS** shows the value **IN_PROGRESS** and **ACTUAL_START_DATE** stores the start timestamp.

- **A new timezone version is available. Please restart to upgrade to timezone version version_number**: If the timezone version on the instance is not the latest, you have the option to upgrade to the latest timezone version.

The following table shows the **DB_NOTIFICATIONS** columns and datatypes.

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>VARCHAR2(128)</td>
<td>Specifies the type of the notification, either a maintenance notification or a timezone version upgrade notification. Valid values are: MAINTENANCE, TIMEZONE VERSION.</td>
</tr>
<tr>
<td>TIME</td>
<td>TIMESTAMP(6) WITH TIMEZONE</td>
<td>Time when the notification entry was added.</td>
</tr>
<tr>
<td>EXPECTED_START_DATE</td>
<td>TIMESTAMP(6) WITH TIMEZONE</td>
<td>Scheduled maintenance start time.</td>
</tr>
</tbody>
</table>
### Column Datatype Description

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPECTED_END_DATE</td>
<td>TIMESTAMP(6) WITH TIME ZONE</td>
<td>Scheduled maintenance end time.</td>
</tr>
<tr>
<td>ACTUAL_START_DATE</td>
<td>TIMESTAMP(6) WITH TIME ZONE</td>
<td>Actual maintenance start time.</td>
</tr>
<tr>
<td>ACTUAL_END_DATE</td>
<td>TIMESTAMP(6) WITH TIME ZONE</td>
<td>Actual maintenance end time.</td>
</tr>
<tr>
<td>MAINTENANCE_PRODUCT</td>
<td>VARCHAR2(128)</td>
<td>Product/component for which maintenance is scheduled/ongoing.</td>
</tr>
<tr>
<td>MAINTENANCE_STATUS</td>
<td>VARCHAR2(128)</td>
<td>Current status of the maintenance.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR2(128)</td>
<td>The notification message details.</td>
</tr>
<tr>
<td>PATCH_ID</td>
<td>VARCHAR2(128)</td>
<td>Patch version. This field is not shown for TIMEZONE VERSION type records.</td>
</tr>
</tbody>
</table>

## View and Manage Customer Contacts for Operational Issues and Announcements

You can view and manage the Autonomous Database customer contacts.

When customer contacts are set, Oracle sends notifications to the specified email addresses for Autonomous Database service-related issues. Contacts in the customer contacts list receive unplanned maintenance notices and other notices, including but not limited to notices for database upgrades and upcoming wallet expiration. When customer contacts are not set the notifications go to the tenancy admin email address associated with the account. Oracle recommends that you set the customer contacts so that the appropriate people receive service-related notifications.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the equilateral triangle next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select an Autonomous Database from the links under the Display Name column.

To view or manage the customer contact email addresses, do the following:

1. On the Autonomous Database Details page, under Maintenance, in the Customer Contacts field, click the Manage link.
   
   This shows the Manage Contacts page. On the Manage Contacts page you can view the email contact list.

2. To add a contact, click Add Contacts.
3. Enter the contact email address.
4. To add additional contacts, click Add Contacts again.
5. Click **Add Contacts** at the bottom of the page to add the new contacts. The lifecycle state changes to Uploading while the customer contact list is updated.

To remove customer contacts, on the Manage Contacts page, do the following:

1. To remove email addresses, select the email addresses to remove.
2. To remove all email addresses, select the top column next to **Email**.
3. Click **Remove**.
4. On the Confirm remove contact page, click **Remove** to confirm. The lifecycle state changes to Uploading while the customer contact list is updated.

To edit customer contacts, on the Manage Contacts page, do the following:

1. To edit email addresses, select the email addresses to edit.
2. To edit all email addresses, select the top column next to **Email**.
3. Click **Edit**.
4. On the Edit Contacts page edit the contacts as needed and click **Save**. The lifecycle state changes to Uploading while the customer contact list is updated.

**Note:**

Subscribe to Oracle Cloud Infrastructure Events to be notified of Autonomous Database events such as when regular maintenance is scheduled or when regular maintenance starts and ends. See Use Autonomous Database Events for more information.

---

## Use Autonomous Database Events

You can use Oracle Cloud Infrastructure Events to subscribe to and be notified of Autonomous Database events. The Oracle Cloud Infrastructure Events allow you to create automation and to receive notifications based on state changes for resources.

**Topics**

- About Events Based Notification and Automation on Autonomous Database
- Get Notified of Autonomous Database Events

## About Events Based Notification and Automation on Autonomous Database

An event could be a resource lifecycle state change or a system event impacting a resource. For example, an event is emitted on Autonomous Database when a backup or restore operation begins or ends. Using Oracle Cloud Infrastructure Events you can subscribe to events. This allows you to receive notifications or to perform other types of automation for events.

You can subscribe to Autonomous Database events as follows:
• Critical Events: See About Critical Events on Autonomous Database for details on the events in the Critical event category.
• Information Events: See About Information Events on Autonomous Database for details on the events in the Information event category.
• Individual Events. See Database Service: Autonomous Database Event Types for the list of Autonomous Database events.

See Overview of Events for complete information on Oracle Cloud Infrastructure Events.

See Get Notified of Autonomous Database Events for information on creating event rules.

About Critical Events on Autonomous Database

Critical events on Autonomous Database are issues that cause disruption to the database.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
</table>
| AdminPasswordWarning | Provides a message that the Autonomous Database ADMIN password is expiring within 30 days or is expired.  
  • If the ADMIN password is within 30 days of being unusable, you receive a warning indicating the date when the ADMIN password can no longer be used.  
  • If the ADMIN password expires and is no longer usable, Autonomous Database reports an event that specifies that the ADMIN user password has expired and must be reset. |
| DatabaseDownBegin    | The Autonomous Database instance cannot be opened, or the services such as high, low, medium, tp, or tpurgent are not started or available.  
  The following conditions do not trigger DatabaseDownBegin:  
  • Operations performed during the maintenance window  
  • Load balancer, network, or backup related issues  
  • A user stopping the instance  
  This event will not be triggered if you are using Autonomous Data Guard and the standby database is not available due to any of these conditions. |
| DatabaseDownEnd      | The database is recovered from the down state, meaning the Autonomous Database instance is opened with its services, following a DatabaseDownBegin event. DatabaseDownEnd is triggered only if there was a preceding DatabaseDownBegin event.  
  The following conditions do not trigger DatabaseDownEnd:  
  • Operations performed during the maintenance window  
  • A user starting the instance  
  If you are using Autonomous Data Guard and the primary database goes down, this triggers a DatabaseDownBegin event. If the system fails over to the standby database, this triggers a DatabaseDownEnd event. |
### Event Description

**DatabaseInaccessibleBegin**
The Autonomous Database instance is using customer-managed keys and the database becomes inaccessible (state shows **Inaccessible**).

The following **conditions** trigger **DatabaseInaccessibleBegin**:
- Oracle Cloud Infrastructure Vault master encryption key is deleted.
- Oracle Cloud Infrastructure Vault master encryption key is disabled.
- Autonomous Database instance is not able to reach the Oracle Cloud Infrastructure Vault.

**DatabaseInaccessibleEnd**
If the Autonomous Database instance recovers from the **Inaccessible** state that generated a **DatabaseInaccessibleBegin** event, when the database state changes to **Available**, the database triggers a **DatabaseInaccessibleEnd** event.

**WalletExpirationWarning**
This event is generated when Autonomous Database determines that a wallet is due to expire in less than six (6) weeks. This event is reported at most once per week. This event is triggered when there is a connection that uses the wallet that is due to expire.

Use the event type **Autonomous Database - Critical** to specify critical events in Event rules.

### About Information Events on Autonomous Database

Information events are operations that provide important details about the database lifecycle on Autonomous Database, such as the time when maintenance begins and maintenance ends, or notifications of connections from a new IP address.

The times shown with these events are in UTC.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AJDNonJsonStorageExceeded</strong></td>
<td>This event is generated when an Autonomous JSON Database has exceeded the maximum storage limit of 20GB of data stored outside of SODA collections. This limit does not apply to data stored in SODA collections or objects associated with SODA collections, such as indexes or materialized views. In addition to this event, an email is sent to the account owner. You must either reduce your usage of non-SODA-related data to below the 20GB limit or promote the Autonomous JSON Database to Autonomous Transaction Processing. See Promote to Autonomous Transaction Processing for more information.</td>
</tr>
<tr>
<td><strong>APEXUpgradeAvailable</strong></td>
<td>This event is generated when you are using Oracle APEX and a new release becomes available.</td>
</tr>
<tr>
<td><strong>DatabaseConnection</strong></td>
<td>This event is generated if a connection is made to database from a new IP address (the connection has not been made from the specified IP address in the last 30 days).</td>
</tr>
</tbody>
</table>
Event | Description
--- | ---
**MaintenanceBegin** | This event is triggered when the maintenance starts and provides the start timestamp for the maintenance (this event does not provide the scheduled start time).

**MaintenanceEnd** | This event is triggered when the maintenance ends and provides the end timestamp for the maintenance (this event does not provide the scheduled end time).

**NewMaintenanceSchedule** | This event is generated when the maintenance date is updated and the new date is shown on the Oracle Cloud Infrastructure Console.

**ScheduledMaintenanceWarning** | This event is generated when the instance is 24 hours from a scheduled maintenance and again when the instance is 1 hour (60 minutes) from the scheduled maintenance.

**Note:**
When Autonomous Data Guard is enabled, any of these events that occur on the standby database do not trigger an Information event.

Use the event type **Autonomous Database - Information** to specify information events in Event rules.

See [View Patch and Maintenance Window Information, Set the Patch Level](#) for information on maintenance windows.

### Get Notified of Autonomous Database Events

Using Oracle Cloud Infrastructure Events you can subscribe to Autonomous Database events individually or in two categories, Critical and Information events.

**Note:**
When you subscribe to an event category, either Critical or Information, you are notified when any events in the category occur. For example, to get notified when the database goes down, subscribe to the Autonomous Database Critical event type.

If you want to subscribe to Critical events, then create an event rule using the **Autonomous Database - Critical** event type. Likewise, if you want to be notified of Information events then create an event rule using the **Autonomous Database - Information** event type.

See [About Events Based Notification and Automation on Autonomous Database](#) for information on Critical and Information events.

See [Getting Started with Events](#) for information on using Oracle Cloud Infrastructure Events, creating event rules, and configuring actions for notifications.
Manage Time Zone File Version on Autonomous Database

For time zone support, Oracle Database uses time zone files that store the list of all time zones. The time zone files for Autonomous Database are periodically updated to reflect the latest time zone specific changes.

Autonomous Database automatically picks up updated time zone files depending on the state of the instance:

- **Stopped**: At the next start operation the update is automatically applied.
- **Available**: After a restart the update is automatically applied.

When a load or import operation results in the following timezone related error, restart your instance and try again:

ORA-39405: Oracle Data Pump does not support importing from a source database with TSTZ version \( n+1 \) into a target database with TSTZ version \( n \).

The restart operation upgrades the time zone file to the latest version available for your database. The columns in your database with the datatype `TIMESTAMP WITH TIME ZONE` are converted to the new time zone version during the restart. See `TIMESTAMP WITH TIME ZONE` Data Type for more information.
Using Standby Databases with Autonomous Database for Disaster Recovery

Autonomous Database uses a feature called Autonomous Data Guard to enable a standby (peer) database to provide data protection and disaster recovery for your Autonomous Database instance.

Topics

• About Standby Databases
• Enable a Standby Database
• Add a Standby Database
• Disable a Standby Database
• Perform a Switchover
• Automatic Failover with a Standby Database
• Manual Failover with a Standby Database
• Events and Notifications for a Standby Database
• Use the API
• Autonomous Data Guard Notes
• Autonomous Data Guard Paired Regions

About Standby Databases

When you enable Autonomous Data Guard the system creates a standby database that is continuously updated with the changes from the primary database. You can enable Autonomous Data Guard with a standby in the current region, a local standby, or with a standby in a different region, a cross-region standby. You can also enable Autonomous Data Guard with both a local standby and a cross-region standby.

Note:

Standby databases incur additional costs. Refer to the relevant Cloud Service Descriptions to learn more:

• Oracle PaaS and IaaS Universal Credits Service Descriptions (PDF)
• Oracle Government IaaS and PaaS Service Descriptions (PDF)

Topics

• Autonomous Data Guard with Local Standby
Autonomous Data Guard with Local Standby

When you enable Autonomous Data Guard with a standby database in the current region, Autonomous Database monitors the primary database and if the primary database goes down, the standby instance automatically assumes the role of the primary instance.

With Autonomous Data Guard enabled with a local standby database, Autonomous Database provides an identical standby database that allows the following, depending on the state of the primary database:

- If your primary database goes down, Autonomous Data Guard converts the standby database to the primary database with minimal interruption. After failover completes, Autonomous Data Guard creates a new standby database for you.
- You can perform a switchover operation, where the primary database becomes the standby database, and the standby database becomes the primary database.

Autonomous Database does not provide access to a standby database in the current region. You perform all operations, such as scaling up the OCPU Count and enabling Auto Scaling on the primary database and Autonomous Data Guard then performs the same actions on the local standby database. Likewise, you only perform actions such as stopping or restarting the database on the primary database.

A local standby database is created in the same region as the primary database (current region). For better resilience, the standby database is provisioned as follows:

- In regions with more than one availability domain, a local standby database is provisioned automatically in a different availability domain than the primary database.
- In regions with a single availability domain, a local standby database is provisioned automatically on a different physical machine than the primary database.

All Autonomous Database features from the primary database are available when the local standby instance becomes the primary after the system fails over or after you perform a switchover operation, including the following:

- **Database Options:** The OCPU Count, Storage, Display Name, Database Name, Auto Scaling, Tags, and Licensing options have the same values after a failover to the standby database or after you perform a switchover.
- **OML Notebooks:** Notebooks and users created in the primary database are available in the standby.
- **APEX Data and Metadata:** APEX information created in the primary database is copied to the standby.
- **ACLs:** The Access Control List (ACL) of the primary database is duplicated for the standby.
- **Private Endpoint**: The private endpoint from the primary database applies to the standby.
- **APIs or Scripts**: Any APIs or scripts you use to manage the Autonomous Database continue to work without any changes after a failover operation or after you perform a switchover.
- **Client Application Connections**: Client applications do not need to change their connection strings to connect to the database after a failover to the standby database or after you perform a switchover.
- **Wallet Based Connections**: You can continue using your existing wallets to connect to the database after a failover to the standby database or after you perform a switchover.

**Autonomous Data Guard with Cross-Region Standby**

When you enable Autonomous Data Guard with a standby database in a different region, if the primary instance goes down, Autonomous Data Guard provides a standby instance that is available to assume the role of the primary instance. The standby database is a replica of the primary database and may be used for recovery in case of failure or when the primary is not available. Enabling Autonomous Data Guard with a cross-region standby provides a solution for disaster recovery in the event an entire region is not available or when the primary database is down for any reason.

Autonomous Data Guard paired regions are remote regions where you can create a cross-region standby database. Autonomous Data Guard allows you to create one remote standby database. See [Autonomous Data Guard Paired Regions](#) for more information on paired regions.

After you enable Autonomous Data Guard with a remote standby, Autonomous Database provides access to the remote standby database from the Oracle Cloud Infrastructure Console. You perform almost 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 cross-region standby database. Autonomous Database provides access to the cross-region standby so that you can perform some operations independently on the remote standby, such as configuring networks and VCNs for private endpoints and tagging to define keys and values that are not replicated between the primary database and the remote standby.

Autonomous Data Guard does not perform automatic failover for a cross-region standby. If the primary database is unavailable and a local standby is unavailable, you can perform a manual failover to make the remote region standby database assume the primary role.

You cannot connect to a remote standby and a remote standby database is not available for read-only operations. You can connect to the remote region database when it assumes the primary role after a switchover or a manual failover.

The following areas have differences for failover and switchover when you failover or switchover from the primary in the primary region to the standby in the remote region, compared to failover or switchover to a local standby:

- **Display Name**: The display name has a "_Remote" extension.
- **OML Notebooks**: After a cross-region switchover or failover, OML notebooks from the primary region are not present in remote region. New OML notebooks can be created in the remote region.
- **Private Endpoint**: You can independently configure and update private endpoints on the remote standby database before failover or switchover. Thus, after a failover or a
switchover, the a private endpoint may be configured differently. Autonomous Database does not keep the networking configuration synchronized from the primary to the standby in the remote region.

VCN Peering and domain forwarding are required for wallets to work across regions, with Autonomous Databases with a private endpoint with Autonomous Data Guard enabled where the primary and the remote database are in different VCNs. See Remote VCN Peering using an RPC and DNS in Your Virtual Cloud Network for information on VCN peering and domain forwarding.

- **APIs or Scripts:** Any APIs or scripts you use to manage the Autonomous Database need to be updated to call the APIs on the remote region's database after a failover operation or after you perform a switchover.

  For best performance and quickest connection time, Oracle recommends that you download a wallet from the remote region database when you use the remote region database as the primary database, after a failover or a switchover.

- **Client Applications:** Client applications may use the single instance wallet containing both primary and standby's connection strings, and do not need to change their wallet to connect to the database, after a failover or switchover to the standby database.

  For best performance and quickest connection time, Oracle recommends that you download a wallet from the remote region database when you use the remote region database as the primary database, after a failover or a switchover.

- **Wallet Based Connections:** You can continue using your existing instance wallets to connect to the database after a failover to the standby database or after you perform a switchover.

  For best performance and quickest connection time, Oracle recommends that you download a wallet from the remote region database when you use the remote region database as the primary database, after a failover or a switchover.

- **Autonomous Database Tools:** The tools have different URLs in the remote region after switchover or failover to the cross-region standby (the tools URLs do not change for a switchover or failover to a local standby):
  - Database Actions
  - Oracle APEX (APEX)
  - Oracle REST Data Services (ORDS)
  - Graph Studio
  - Oracle Machine Learning Notebooks
  - Oracle Machine Learning User Management

- **Oracle Cloud Infrastructure Object Storage Usage:** After you failover or switchover from the primary in the primary region to the standby in the remote region, the credentials and the URLs that provide access to Object Storage in the primary region continue to work as they did before the failover or the switchover, providing access to the following:
  - Manual backups
  - External tables
  - External partitioned tables
  - External hybrid partitioned tables
Note:

This applies when the primary region's Object Storage is available. For rare scenarios when the Object Storage in the primary region is not available, Oracle recommends having Object Storage backups or replication to a different region. If the primary region's Object Storage is not available, you may update your user credentials and parameters that set URLs for Object Storage so that the parameters specify values to access the available region's Object Storage. See Using Replication for more information.

Autonomous Data Guard Database Region and Role

After you enable Autonomous Data Guard with a cross-region standby database, Autonomous Data Guard specifies each database based on its region as the primary region database and the remote region database and each database has a designated role: primary or standby.

The Autonomous Database Information page shows the region in the Region field. The Region field shows one of two values, Primary or Remote, based on the database's role when you enable Autonomous Data Guard, and this value does not change.

The role specifies the current state of a database, primary or standby, and this value changes after you perform a switchover or a manual failover. You can view the Autonomous Database role in the icon that shows next to the display name on the Autonomous Database Information page:

![DOC SALES Primary](image)

![DOC SALES_Remote Standby](image)

The Autonomous Database Information page also shows the role in Autonomous Data Guard area in the Role field.

Thus, after you enable Autonomous Data Guard with a cross-region standby, the Region field for the database shows Primary and the Role field also Primary. After a switchover, the same database shows the Region as Primary and the Role as Standby.

Note:

Oracle recommends using the primary region when available. The remote region is available for testing and for disaster recovery as required. If disaster recovery is required, you can use the remote region to continue operations and return to using the primary region when it is available.

You can see the peer Autonomous Database details for both the local and the remote peer database. To see this information, on the Autonomous Database Information page under Resources, select Autonomous Data Guard. For a local standby database, the database has
the same display name in the **Peer Autonomous Database** column. For a remote standby database, the database has the same display name with a "Remote" extension and provides a link to access the remote database. The **Peer Role** column shows the role for the peer database, either **Primary** or **Standby**.

![Autonomous Data Guard](image)

**Autonomous Data Guard Failover and Switchover with Cross-Region Standby and No Local Standby**

With Autonomous Data Guard enabled with a cross-region standby without a local standby, you have the following options:

- If your primary database goes down, you can manually failover to the remote standby database.
- You can perform a switchover operation, where the primary region database becomes the standby database, and the remote region standby database becomes the remote region primary database.

**Autonomous Data Guard Failover and Switchover with Local Standby and Cross-Region Standby**

With Autonomous Data Guard enabled with a local standby database and a cross-region standby database, Autonomous Database provides a local standby database and a cross-region standby database.

With both a current region and a remote region standby database, depending on the state of the primary region database, you have the following options:

- If your primary database goes down and the local standby database is available, Autonomous Data Guard automatically performs failover to convert the local standby database to the primary database with minimal interruption. After failover completes, Autonomous Data Guard creates a new local standby database for you. If automatic failover is not possible, you have the option to perform a manual failover.
  
  Autonomous Data Guard continues to use the same cross-region standby.

- If your primary database goes down and the local standby database is not available, you can perform a manual failover to the cross-region standby database.

  In this case, the remote standby database becomes the primary database. After failover completes, Autonomous Data Guard does not create a new local standby database for you. In this case, the remote region assumes the primary role but does not have a local standby.
• You can perform a switchover operation, where the primary database becomes the local standby database, and the local standby database becomes the primary database.

Autonomous Data Guard continues to use the same cross-region standby.

This option is not available if you are using the remote region database as your primary database (that is **Region** is Remote and **Role** is Primary). For example, after you perform a switchover or a failover to the remote region.

• You can perform a switchover operation, where the remote region standby database becomes the remote region primary database and the primary region primary database becomes the primary region standby database.

A switchover changes the primary region database to the Standby role. If you perform a switchover two times, the primary region database returns to the Primary role.

---

**Note:**

When you enable Autonomous Data Guard with both a local and a cross-region standby, Autonomous Data Guard does not provide a local standby while the remote region instance operates in the Primary role. Using the remote region in the Primary role is intended for use while the primary region is unavailable or for testing (a temporary scenario). After the primary region database returns to the Primary role, the local Standby will be available.

---

**Autonomous Data Guard Database Cross Region Backup and Restore**

After you enable Autonomous Data Guard with a cross-region standby database, backup and restore from backup is handled as follows:

• If the primary database is restored from a backup, a new remote region standby instance is created from the restored instance.

• Automatic Backups and Manual backups are only taken on the primary database (the database showing **Role**: Primary). For example, after a switchover or failover to the remote region, the database in the remote region assumes the primary role and starts to perform automatic backups. The primary region database with role Standby no longer takes backups. If you switch back so the primary region's database role becomes Primary, then the database in the primary region starts taking backups again.

• You cannot restore or clone from a backup when either the primary region database or the remote region database is in the Standby role. Backups are only taken on the database in the Primary role, and the restore operation is not available from the Oracle Cloud Infrastructure Console on the Standby database.

---

**Autonomous Data Guard and Primary Region Wallets**

After you enable Autonomous Data Guard with a remote standby, on the primary database download a new instance wallet. The instance wallet file you download from the primary database contains connection strings for both the primary region and the remote region database.

The order of the connection strings in the instance wallet file impacts the database connection time. For best performance, use the wallet file downloaded from the region in which the current Primary instance resides.
When you download a regional wallet, the wallet only contains the connection strings for the primary or standby database that lies in that same region as the downloaded regional wallet. Regional wallets do not contain the connection strings for remote databases.

See Download Client Credentials (Wallets) for more information.

**Autonomous Data Guard Recovery Time Objective (RTO) and Recovery Point Objective (RPO)**

Autonomous Data Guard monitors the primary database and if the instance goes down, then the local standby instance assumes the role of the primary instance according to the Recovery Time Objective (RTO) and Recovery Point Objective (RPO). If the local standby instance is not available and you have enabled a cross-region standby, you can manually failover to a cross-region standby.

The RTO is the maximum amount of time required to restore database connectivity to a standby database after a manual or automatic failover is initiated. The RPO is the maximum duration of potential data loss, in minutes, on the primary database.

When Autonomous Data Guard is enabled with a local standby database, the RTO and RPO numbers are as follows:

- **Automatic Failover or Switchover**: the RTO is two (2) minutes and RPO is zero (0).
- **Manual Failover**: the RTO is two (2) minutes and RPO is up to one (1) minute.

When Autonomous Data Guard is enabled with a cross-region standby database, the RTO and RPO numbers for failover to the cross-region standby are as follows:

- **Switchover**: the RTO is fifteen (15) minutes and RPO is zero (0).
- **Automatic Failover**: Not available
- **Manual Failover**: the RTO is fifteen (15) minutes and RPO is up to one (1) minute.

See [Perform a Switchover](#) for details on switchover.

See [Automatic Failover with a Standby Database](#) for details on automatic failover.

See [Manual Failover with a Standby Database](#) for details on manual failover.

**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 a Standby Database](#) for details.

- **Add Standby Database**: After you enable Autonomous Data Guard with either a local (current region) standby or a cross-region standby database (remote), you can add a second standby database. If the standby database you enable first is a remote database, you can add a local (current region) standby database. If the standby database you enable first is a current region standby database, you can add a cross-region standby database.
See Add a Standby Database for details.

• **Disable:** If Autonomous Data Guard is enabled, you can disable Autonomous Data Guard. Disabling Autonomous Data Guard terminates the standby database. If you have both a local standby database (current region), and a cross-region standby database (remote), you disable each standby individually.
  See Disable a Standby Database for details.

• **Switchover:** When Autonomous Data Guard is enabled, switchover changes the roles of the primary and the standby, the standby database becomes the primary, and the primary database becomes the standby. If you have both a local standby database (current region), and a cross-region standby database (remote), you can choose to switchover either the local standby or the remote standby.
  See Perform a Switchover for details.

• **Manual Failover:** If the primary database is not available, you can perform a manual failover to change roles to make a standby database the primary database:
  – If a local standby is available, you can manually failover to the local standby (you do not have the option to failover to a remote region standby if a local standby is available).
  – If a local standby is not enabled or was enabled with Autonomous Data Guard and is not available, you have the option to manually failover to a remote standby.
  See Manual Failover with a Standby Database for details.

• **Terminate:** If you want to terminate the primary instance, select More Actions and Terminate. Terminating the primary instance also terminates the standby database. If you have both a local standby database (current region), and a cross-region standby database (remote), this terminates both the local standby and the remote standby.

**Autonomous Database Standby Database State**

Autonomous Database provides information about Autonomous Data Guard state on the Autonomous Database Details page.

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 **Role** field shows the role of the current database, as follows:

• When using Autonomous Data Guard with a local standby, the Oracle Cloud Infrastructure Console shows the **Role** field value **Primary**. Autonomous Database does not provide access to a local standby database.

• When using Autonomous Data Guard with a cross-region standby, the Oracle Cloud Infrastructure Console shows the **Role** field value **Primary** if you are viewing the primary database and shows **Standby** if you are viewing the details for the standby database.

• **Switchover:** Provides a link so that you can perform a switchover operation.

• **Failover:** When the primary database is not available and you have a local standby and automatic failover was not successful, the failover link allows you initiate a manual failover.
When the primary database is not available and you have a cross-region standby and failover to a local standby is not possible, the failover link allows you initiate a manual failover to the remote standby database.

To view the peer Autonomous Database information, under Resources click Autonomous Data Guard. This area lists the peer autonomous database information. The State column shows the state of a standby database, as follows:

- **Provisioning**: This state shows when you enable Autonomous Data Guard and indicates that a standby database is provisioning (until the standby database state changes to Standby).
- This state shows after a failover to a local standby when a local standby database is being recreated.
- This state shows if a restore from backup operation is performed on the primary database, the local standby is recreated and the State column shows Provisioning.

- **Standby**: Indicates that a standby is available and ready for either a switchover or a failover operation.

**Note:**

When the standby database is stopped, the standby state shows **Standby**. A standby database never shows the **Stopped** state.

- **Role Change in Progress**: Indicates a failover or switchover operation started.

**Autonomous Data Guard Events with Autonomous Database**

You can use Oracle Cloud Infrastructure events to respond when Autonomous Database changes its state due to an Autonomous Data Guard related event such as a failover or switchover operation.

Autonomous Database events include the following:

- Begin disable Autonomous Data Guard
- Begin enable Autonomous Data Guard
- Begin failover
- Begin switchover
- End disable Autonomous Data Guard
- End enable Autonomous Data Guard
- End failover with failover result of success or failure.
- End switchover with switchover result of success or failure.

Based on events you can perform actions or send notifications. See Events and Notifications for a Standby Database for more information on using events and producing notifications.
Enable a Standby Database

You can enable Autonomous Data Guard when Autonomous Database is available (Lifecycle State shows **Available**).

**Note:**

To enable Autonomous Data Guard you must have adequate available resources and by enabling Autonomous Data Guard you must not exceed your Tenancy or compartment limits for CPU and Storage.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display Name** column.

To enable Autonomous Data Guard do the following:

1. On the **Autonomous Database Details** page, under Autonomous Data Guard, click **Enable** in the **Status** field.

2. In the **Region** field, select a region.
a. To create a local standby database, select the Current Region. To create a cross-region standby database, select a different region.

The region list shows the current region for a local standby and the available remote regions where you can create a cross-region standby.

b. If you select a different region, Autonomous Database shows the Select a compartment list. From this list, choose a compartment for the cross-region standby.

Note:

When you enable Autonomous Data Guard, the list of available Standby databases only shows a remote region if your tenancy is subscribed to the remote region (you must be subscribed to the paired remote region for the region where you are enabling Autonomous Data Guard).

3. Click Enable Autonomous Data Guard.

The Autonomous Database Lifecycle State changes to Updating. In the Resources area with Autonomous Data Guard selected, the State field shows Provisioning.

After some time, the Lifecycle State shows Available and the standby database provisioning continues.

Note:

While you enable Autonomous Data Guard the primary database is available for read/write operations. There is no downtime on the primary database.

When provisioning completes, on the Autonomous Database Details page under Autonomous Data Guard, the Role field shows Primary with a link Switchover that you can click to perform a switchover.

If you enable a remote standby, there is an additional Region field that shows Primary.
If you enable a cross-region standby, the standby database created in the remote region has the same display name as the primary database, with the extension, "_Remote". The standby database display name and a link to the standby database shows under the **Peer Autonomous Database** column when you click **Autonomous Data Guard** under **Resources**.

**Notes for enabling Autonomous Data Guard:**

- Autonomous Database generates the Enable Autonomous Data Guard work request. To view the request, under **Resources** click **Work Requests**.
- After you enable Autonomous Data Guard with a remote standby, download a new instance wallet. The instance wallet file you download from the primary database contains connection strings for both the primary region and the remote region database. The same instance wallet works after you switchover or failover to the remote region standby.
  
  The order of the connections strings in the instance wallet file impacts the database connection time. For best performance, use an instance wallet file downloaded from the region in which the current Primary instance resides.
  
  - While you enable Autonomous Data Guard, when the **Lifecycle State** field shows **Updating**, the following actions are disabled for the primary database:
    - Move Resource. See **Move an Autonomous Database to a Different Compartment** for information on moving an instance.
    - Stop. See **Stop Autonomous Database** for information on stopping an instance.
    - Restart. See **Restart Autonomous Database** for information on restarting an instance.
    - Restore. See **Restore and Recover your Autonomous Database** for information on restoring.

**Add a Standby Database**

When you enable Autonomous Data Guard you create one standby database, either in the current region (local), or in a remote region (cross-region). After you enable Autonomous Data Guard, you can add a second standby database.

When you enable Autonomous Data Guard with a current region (local) standby database, you can add a second standby database that is a remote (cross-region) standby database. When you enable Autonomous Data Guard with a remote standby database (cross-region), you can add a second standby database that is a current region (local) standby database.

Perform the following prerequisite steps as necessary:

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

• On the Autonomous Databases page select your Autonomous Database from the links under the **Display Name** column.

To add a standby database, do the following:

1. On the **Autonomous Database Details** page, in the **Resources** area select **Autonomous Data Guard**.

2. Click **Add Standby Database**.

   ![Add Standby Database](image)

If you select a remote region for your standby, Autonomous Database shows the **Select a compartment** list. From this list, choose a compartment.

3. Click **Add Standby Database**.

   The Autonomous Database Lifecycle State changes to **Updating**. In the **Resources** area the number next to **Autonomous Data Guard** shows that you now have two standby databases (2), one local and one remote, and the **State** field shows **Provisioning** for the new standby database.

   After some time, the **Lifecycle State** shows **Available** and the standby database provisioning continues.

   **Note:**

   While you add a new standby database, the primary database is available for read/write operations. There is no downtime on the primary database.

---

**Chapter 35**

**Add a Standby Database**

**35-14**
Notes for adding a second standby database with Autonomous Data Guard:

- Autonomous Database generates an Enable Autonomous Data Guard work request. To view the request, under Resources click Work Requests.
- After you add an Autonomous Data Guard with a remote standby, download a new instance wallet. The instance wallet file you download from the primary database contains connection strings for both the primary region and the remote region database. The same instance wallet works after you switchover or failover to the remote region standby.

The order of the connections strings in the instance wallet file impacts the database connection time. For best performance, use an instance wallet file downloaded from the region in which the current Primary instance resides.

- While you add a standby database with Autonomous Data Guard, when the Lifecycle State field shows Updating, the following actions are disabled for the primary database:
  - Move Resource. See Move an Autonomous Database to a Different Compartment for information on moving an instance.
  - Stop. See Stop Autonomous Database for information on stopping an instance.
  - Restart. See Restart Autonomous Database for information on restarting an instance.
  - Restore. See Restore and Recover your Autonomous Database for information on restoring.

### Disable a Standby Database

Descries the steps to disable Autonomous Data Guard.

#### Note:

Disabling Autonomous Data Guard terminates the standby database. If you later enable Autonomous Data Guard, the system creates a new standby database.

To disable Autonomous Data Guard you individually disable the local standby database or terminate the cross-region standby database. If you have both a local standby and a remote standby, you first disable one and then the other. The order you disable multiple standby databases is not significant, either local first or remote first.

- Disable a Local Standby Database
- Terminate a Cross-Region (Remote) Standby Database
- Verify Autonomous Data Guard is Disabled

### Disable a Local Standby Database

Describes the steps to disable a local standby database.

Perform the following prerequisite steps as necessary:

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

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

To disable a local standby database do the following:

1. On the Autonomous Database Details page, under Autonomous Data Guard, in the **Status** field, click **Disable**.

2. In the Disable Autonomous Data Guard dialog, select the local standby database.

3. In the Disable Autonomous Data Guard dialog, enter the Autonomous Database name.

4. In the Disable Autonomous Data Guard dialog, click **Disable Autonomous Data Guard**.

   While the local standby database is terminating the **Lifecycle State** changes to **Updating**.

   Autonomous Database generates the Disable Autonomous Data Guard work request. To view the request, under **Resources** click **Work Requests**.

After disabling the local standby, there are two possible results:

- If there is no remote standby database, disabling the local standby database terminates the standby database and disables Autonomous Data Guard. See **Verify Autonomous Data Guard is Disabled** for more information.

- If there is a cross-region standby database, disabling the local standby database terminates the local standby database but does not disable Autonomous Data Guard. In this case, to disable Autonomous Data Guard you must terminate the cross-region standby database. See **Terminate a Cross-Region (Remote) Standby Database** for more information.
Terminate a Cross-Region (Remote) Standby Database

Describes the steps to terminate a cross-region standby database.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the Display Name column.

To terminate a cross-region (remote) standby database:

1. On the Primary Region Autonomous Database, on the Autonomous Database Details page under Resources, select Autonomous Data Guard.
2. Access the remote region Autonomous Database instance.
   - On the Primary Region Autonomous Database instance the Autonomous Data Guard information area shows the Peer Autonomous Database. The remote standby database has the same name as the Primary Region database, with an "Remote" extension.
   - Under Peer Autonomous Database, click the remote standby database to access the cross-region Autonomous Database instance.
3. On the remote Autonomous Database instance, on the Details page, from the More Actions drop-down list, select Terminate.

For example:

![Terminate Autonomous Database dialog box](image-url)
4. On the **Terminate Autonomous Database** page enter the database name to confirm that you want to terminate the cross-region standby database.

5. **Click** **Terminate Autonomous Database**.

   While the standby database is terminating, the **Lifecycle State** changes to **Terminating**.

After you terminate the remote standby database, there are two possible results:

- If there is no local standby database, terminating the cross-region standby database disables Autonomous Data Guard. See [Verify Autonomous Data Guard is Disabled](#) for more information.
- If there is a local standby database, terminating the cross-region standby database terminates the remote database but does not disable Autonomous Data Guard. In this case, to disable Autonomous Data Guard you must disable the local standby database. See [Disable a Local Standby Database](#) for more information.

There are limitations for disabling when Autonomous Data Guard includes a cross-region standby, as follows:

- A standby database in the remote region cannot be disabled from the primary database. If you click **Disable** on the primary and select a remote standby database, the system shows a message indicating that a cross-region standby database must be terminated from the remote region. The message includes a link you can use to access the remote standby instance.

For example:

![Disable Autonomous Data Guard](image.png)

- Disabling a database in the **Remote** region is not allowed when the standby in the remote region is running in the **Primary** role. Thus, after a switchover or a failover, if the database with the **Primary** role is in the **Remote** region, **Disable** is not active. When the **Primary** region database is in the **Standby** role, you must first switchover before you terminate the standby database.

- When Autonomous Data Guard is enabled with a cross-region standby database, you must terminate the cross-region standby in the **Remote** region before you
terminate the primary role database. If you attempt to terminate the primary, the system shows the following message:

In this case, after you terminate the cross-region (remote) standby you can terminate the database in the Primary region.

- You cannot terminate the Primary role database when it is running in the Remote region. A switchover or a failover operation is not meant to be permanent. In this case, to disable Autonomous Data Guard you take the following steps:
  1. Perform a switchover so that the Primary role database is running in the Primary region.
  2. After the switchover, terminate the standby database in the Remote region. See Terminate a Cross-Region (Remote) Standby Database for more information.
  3. If there is a local standby database, disable the local standby database. See Disable a Local Standby Database for more information.

After Autonomous Data Guard is disabled, terminate the database. See Terminate an Autonomous Database Instance for more information.

Verify Autonomous Data Guard is Disabled

Describes the steps to verify Autonomous Data Guard is disabled.

To verify that Autonomous Data Guard is disabled:

1. On the Autonomous Database Details page under Resources, select Autonomous Data Guard.
2. Verify the standby database count is 0 and that the Autonomous Data Guard area shows None Available.
If a local or a cross-region standby database remains and you want to terminate it, individually terminate the local or the cross-region standby:

- Disable a Local Standby Database
- Terminate a Cross-Region (Remote) Standby Database

Perform a Switchover

When you perform a switchover the primary database becomes the standby database, and the standby database becomes the primary database, with no data loss.

A switchover is typically done to test failover to the standby database for audit or certification reasons, or to test your application's failover procedures when Autonomous Data Guard is enabled.

For switchover to a standby database, the Oracle Cloud Infrastructure Console on the database with the **Primary** role shows a **Switchover** link in the **Role** field when both the primary database and a standby database are available. You can perform a switchover when the primary database **Lifecycle State** shows **Available** or **Stopped** and a standby database is available (the **State** field shows **Standby**).

To see the standby database status, under **Resources** click Autonomous Data Guard and for the standby database listed in the **Peer Autonomous Database** column, check that the **State** field shows **Standby**.

Using the Autonomous Database API, you can initiate the switchover operation at any time. See **Use the API** for more information.

Perform the following prerequisite steps as necessary:

**To perform switchover to a local standby, access the Primary database (Role: Primary) from the Oracle Cloud Infrastructure Console:**

- Open the Oracle Cloud Infrastructure Console by clicking the ![Oracle Cloud](next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display Name** column.

**To perform switchover to a remote standby, access the Standby database (Role: Standby):**

> **Note:**
> For a cross-region switchover you must initiate the switchover from the Standby database.

You have several options to access the Standby database:

- Select the remote region in Oracle Cloud Infrastructure Console and then select the Standby database.
• Access the Primary as above, and then from the Primary database you can access the Standby from the Oracle Cloud Infrastructure Console by selecting Autonomous Data Guard under Resources and clicking the link for the standby database in the Peer Autonomous Database column.

To perform a switchover, do the following:

1. On the Autonomous Database Details page, under Autonomous Data Guard, in the Role field, click Switchover.
   
   As an alternative, to initiate a switchover you can select More Actions and Switchover.

2. In the Confirm Manual Switchover to Standby dialog, enter the database name to confirm that you want to switch over.

   
   When concurrent operations such as scaling or creating a manual backup are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

   The database Lifecycle State changes to Updating. To see the state of the peer database, under Resources click Autonomous Data Guard where the State column shows Role Change in Progress.

When the switchover completes, Autonomous Data Guard does the following:

• The display name shows the Standby indicator (for a cross-region switchover).

• The Autonomous Data Guard resource information is updated to reflect the switchover. Select Autonomous Data Guard under Resources to see the updated information.

• Autonomous Database reports the time of the last switchover when you hover over the in the Role field.

Notes for Autonomous Data Guard switchover:

• For a cross-region switchover you must initiate the switchover from the Standby database.

• During the switchover most of the actions on the Oracle Cloud Infrastructure Console are not available and the Autonomous Database Information page shows the Lifecycle State with the value Updating.

• The switchover operation keeps the original state of the Primary database. If the Primary database is stopped when you perform a switchover, the Primary database is stopped after the switchover.

• Autonomous Database generates the Switchover Autonomous Database work request. To view the request, under Resources click Work Requests.

• After a switchover or failover to the Standby, the Standby becomes the Primary and the graphs on the Database Monitor in Database Actions and the Oracle Cloud Infrastructure Metrics display information about the Primary database. The graphs and metrics do not contain information about the database that was the Primary before the switchover or failover operation.

• When you enable Autonomous Data Guard with both a local and a cross-region standby, Autonomous Data Guard does not provide a local standby while the remote region instance operates in the Primary role. Using the remote region in the Primary role is intended for use while the primary region is unavailable or for testing (a temporary
After the primary region database returns to the Primary role, the local Standby will be available.

- You cannot cancel a cross-region switchover operation after the switchover begins and the State shows Role Change in Progress. Your options are:
  - Try or retry a switchover or a failover until the operation succeeds.
  - File a service request at Oracle Cloud Support or contact your support representative.

### Automatic Failover with a Standby Database

After you enable Autonomous Data Guard, the system monitors the primary instance and automatically fails over to a local standby database in certain scenarios. If automatic failover is not possible, you have the option to perform a manual failover. Automatic failover does not apply to a cross-region standby.

**Automatic Database** automatically fails over to a local standby database as follows:

- When the primary database becomes unavailable and users are not able to connect, Autonomous Data Guard automatically fails over to a local standby database if a local standby database is available, based on the Recovery Time Objective (RTO) and Recovery Point Objective (RPO) targets. See Autonomous Data Guard Recovery Time Objective (RTO) and Recovery Point Objective (RPO) for more information.

- Autonomous Data Guard performs automatic failover to a local standby database when a local standby database is available and the system can guarantee zero data loss, as specified by the Recovery Point Objective (RPO) target. If this target cannot be met then automatic failover does not succeed and you have the option to perform manual failover. See Autonomous Data Guard Recovery Time Objective (RTO) and Recovery Point Objective (RPO) for more information.

- After automatic failover to a local standby database completes, Autonomous Database creates a new local standby database for you.

  Autonomous Data Guard for a particular standby, either local or remote, is not enabled while the system is provisioning the new standby database. After Autonomous Data Guard completes the provisioning step for the standby database and it becomes available, you then have a new standby database with Autonomous Data Guard enabled on it.

- After automatic failover completes, Autonomous Database reports the time of the last failover when you hover over the icon in the Role field.

If the primary database has failed or is unreachable and the conditions for Autonomous Data Guard automatic failover have not been met, then Oracle Cloud Infrastructure console shows a banner indicating that automatic failover did not succeed, provides a reason, such as possible data loss, and provides a link for you to initiate manual failover. See Manual Failover with a Standby Database for more information.

**Note:**

Autonomous Data Guard automatic failover is disabled when the Lifecycle State is either: Restore in Progress or Upgrading.
Manual Failover with a Standby Database

When Autonomous Data Guard cannot automatically fail over to a local Standby database, if a local Standby database is available you can perform a manual failover to make the local Standby database the Primary database. If a cross-region Standby is available, you can perform a switchover to make the cross-region Standby database the Primary database and if the switchover fails, you can initiate a manual failover to the cross-region Standby. It is possible for data loss to occur with a manual failover.

Depending on how you enable Autonomous Data Guard, there are different steps to perform a manual failover to a Standby database:

- **When you enable Autonomous Data Guard with just a cross-region (remote) Standby:**
  When you enable a cross-region Standby and do not enable a local Standby, after you attempt a cross-region switchover and if the switchover fails, on the Standby database the Oracle Cloud Infrastructure console shows a failover link in the Role field that you can click to initiate a manual failover. Using the API, you can initiate manual failover at any time. See Use the API for information on using the API.

- **When you enable Autonomous Data Guard with a local Standby:**
  When you enable a local Standby and automatic failover is not successful, Oracle Cloud Infrastructure console shows a banner with information about why the automatic failover was not successful and Oracle Cloud Infrastructure console shows a failover link in the Role field that you can click to initiate a manual failover to the local Standby. The failover link only shows when the Primary database in the primary region is unavailable and a Standby database is available. That is, the Primary database Lifecycle State field shows Unavailable and the local Standby database is available. Using the API, you can initiate manual failover at any time. See Use the API for information on using the API.

  To see the Standby database status, under Resources click Autonomous Data Guard and for the Standby database listed in the Peer Autonomous Database column check that the State field shows Available or Stopped.

- **When you enable Autonomous Data Guard with both a local Standby and a cross-region (remote) Standby:**
  With Autonomous Data Guard enabled with both a local Standby and a cross-region Standby, when automatic failover is not successful and the local Standby database is available, Oracle recommends that you attempt a manual failover to the local Standby first (not to the remote region Standby).

  If a local Standby is unavailable or a manual failover to the local Standby fails, you can perform a manual switchover to the cross-region Standby. If the switchover to the cross-region Standby fails, on the Standby database the Oracle Cloud Infrastructure console shows a failover link in the Role field that you can click to initiate a manual failover to the Standby database.

  When you initiate a manual failover Autonomous Data Guard fails over to the Standby database based on the Recovery Time Objective (RTO) and Recovery Point Objective (RPO) targets. See Autonomous Data Guard Recovery Time Objective (RTO) and Recovery Point Objective (RPO) for more information.

  Perform the following prerequisite steps as necessary:

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

On the Autonomous Databases page select your Autonomous Database from the links under the Display Name column.

To initiate a manual failover to a cross-region Standby, do the following:

1. On the Standby database, perform a switchover. See Perform a Switchover for details.

2. If the switchover attempt in Step 1 fails, on the Standby database the Role field shows a Failover link. On the Standby database, click the Failover link.

   This shows the Confirm Manual Failover to Standby dialog, along with information on possible data loss that may result if you perform the manual failover to the Standby database.

3. In the Confirm Manual Failover to Standby dialog, enter the Autonomous Database name to confirm that you want to failover.


   When concurrent operations such as scaling or creating a manual backup are active, the confirmation also confirms either pausing or canceling the concurrent operation. See Concurrent Operations on Autonomous Database for more information.

To initiate a manual failover when the Primary database is unavailable and the local Standby is available, do the following:

1. On the Details page, under Autonomous Data Guard, in the Role field, click Failover.

   This shows the Confirm Manual Failover to Standby dialog, along with information on possible data loss that may result if you perform the manual failover to standby.

2. In the Confirm Manual Failover to Standby dialog, enter the Autonomous Database name to confirm that you want to failover.


   When the failover completes, Autonomous Data Guard does the following:
• For a failover to the local Standby, Autonomous Data Guard creates a new local Standby database after the failover. Autonomous Data Guard is not enabled when the system is provisioning the new Standby database and the Role field shows Provisioning. After Autonomous Data Guard completes the provisioning step, then you have a new local Standby database and Autonomous Data Guard is enabled.

• After a manual failover operation completes, you can see any data loss associated with the manual failover in the message on the Oracle Cloud Infrastructure console banner and if you hover over the.Redis in the Role field. The manual failover data loss is specified in minutes.

• When you enable Autonomous Data Guard with both a local and a cross-region standby, Autonomous Data Guard does not provide a local standby while the remote region instance operates in the Primary role. Using the remote region in the Primary role is intended for use while the primary region is unavailable or for testing (a temporary scenario). After the primary region database returns to the Primary role, the local Standby will be available.

Events and Notifications for a Standby Database

You can use Oracle Cloud Infrastructure Events to be notified of and to specify rules to automatically respond to Autonomous Data Guard operations.

Note:
When Autonomous Data Guard is enabled with a remote standby, events are only sent to the instance with the Primary role.

<table>
<thead>
<tr>
<th>Event Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable Data Guard Begin</td>
<td>com.oraclecloud.databaseservice.disableautonomousdataguard.begin</td>
</tr>
<tr>
<td>Disable Data Guard End</td>
<td>com.oraclecloud.databaseservice.disableautonomousdataguard.end</td>
</tr>
<tr>
<td>Enable Data Guard Begin</td>
<td>com.oraclecloud.databaseservice.enableautonomousdataguard.begin</td>
</tr>
<tr>
<td>Enable Data Guard End</td>
<td>com.oraclecloud.databaseservice.enableautonomousdataguard.end</td>
</tr>
<tr>
<td>Manual Failover Begin</td>
<td>com.oraclecloud.databaseservice.failoverautonomousdatabase.begin</td>
</tr>
<tr>
<td>Manual Failover End</td>
<td>com.oraclecloud.databaseservice.failoverautonomousdatabase.end</td>
</tr>
<tr>
<td>Switchover Begin</td>
<td>com.oraclecloud.databaseservice.switchoverautonomousdatabase.begin</td>
</tr>
</tbody>
</table>
### Event Description and Type

<table>
<thead>
<tr>
<th>Event Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switchover End</td>
<td>com.oraclecloud.databaseservice.switchover自主database.end</td>
</tr>
</tbody>
</table>

See the following Oracle Cloud Infrastructure topics for information on events and notifications:

- Overview of Events
- Autonomous Database Event Types
- Notifications Overview

### Use the API

For information about using the API and signing requests, see REST APIs and Security Credentials.

For information about SDKs, see Software Development Kits and Command Line Interface.

**Use these API operations to manage Autonomous Data Guard:**

- To enable or disable Autonomous Data Guard, use UpdateAutonomousDatabase.
- To initiate a manual failover operation, use FailOverAutonomousDatabase.
- To initiate a switchover operation, use SwitchOverAutonomousDatabase.

**Use these Terraform APIs to manage Autonomous Database resources:**

For information about Terraform, see Terraform Provider and for information about Terraform APIs, see Data Source: oci_database_autonomous_database.

### Autonomous Data Guard Notes

Note the following for using Autonomous Database with Autonomous Data Guard:

- You cannot connect to a Standby database until it is made the Primary by a failover or a switchover. Thus, a Standby database cannot be opened for read-only access and cannot be used to offload queries from a Primary database.
- Autonomous Data Guard is not available with Always Free Autonomous Databases.
- Autonomous Database does not provide access to a local Standby database:
  - You perform all operations, such as scaling up the OCPU Count and enabling Auto Scaling on the Primary database and Autonomous Database performs the same actions on the local Standby database. Likewise, you only perform actions such as stopping or restarting the database on the Primary database.
  - A local Standby database is not available for use as a read-only database.
- Autonomous Database provides access to a remote Standby database:
You perform most operations, such as scaling up the OCPU Count and enabling Auto Scaling on the Primary database and Autonomous Database performs the same actions on the remote Standby database. Likewise, you only perform actions such as stopping or restarting the database on the Primary database.

You can perform certain operations, such as configuring private endpoints on a remote Standby database.

A remote Standby database is not available for use as a read-only database.

Wallets are disabled on a remote Standby database and you cannot rotate a remote Standby database instance wallet.

- The Number of OCPUs allocated graph and the CPU utilization graph on the Database Monitor in Database Actions displays the OCPUs allocated and the CPUs utilization for the Primary database. These graphs do not include information about a local Standby database or about a remote Standby database.

The CPU Utilization metrics on the Oracle Cloud Infrastructure Console metrics page display the CPU utilization for the Primary database. Other metrics on this page also apply to the Primary database. These metrics do not include information about the local Standby database or remote Standby database.

- After a switchover or failover to the Standby, the Standby becomes the Primary and the graphs on the Database Monitor in Database Actions and the Oracle Cloud Infrastructure Console metrics page display information about the Primary database. The graphs and metric do not contain information about the database that was the Primary before the switchover or failover operation.

- Automatic Failover to a local Standby is disabled during a Restore in Progress operation.

- Automatic Failover to a local Standby disabled when Upgrading a Database.

- When the Lifecycle State field for the Primary database shows Stopped, the Standby database is also stopped. You may still perform a switchover when the Primary database is Stopped.

Cross-Region Autonomous Data Guard Notes

The following are restrictions and limitations when you enable Autonomous Data Guard with a remote Standby database:

- To disable Autonomous Data Guard with a cross-region standby database, you terminate the remote Standby database. See Terminate a Cross-Region (Remote) Standby Database for more information.

- You must use Oracle-managed encryption keys with Autonomous Data Guard with a remote Standby database. Customer-managed encryption keys are not supported with cross-region Autonomous Data Guard.
  - Switching to customer-managed keys is not allowed when Autonomous Data Guard is enabled with a remote Standby.
  - Enabling Autonomous Data Guard with a remote Standby is not allowed if the Autonomous Database is using customer-managed keys.

- When a private endpoint is enabled or disabled on the Primary database, any previously configured Access Control List (ACL) on the Standby is enabled and the values are cleared. You must reset and verify the ACL on the Standby database after you disable a private endpoint on the Primary.
• Oracle Data Safe can be enabled on a database that has a cross-region Standby database enabled, but it only monitors the database within its region, and cannot monitor the standby in the event of a switchover or a failover.

• When you allow TLS authentication for the Primary database, Autonomous Data Guard enables TLS authentication in the cross-region Standby. Thus, when Autonomous Data Guard is enabled with a remote Standby, you can only allow TLS connections on the Primary if both the Primary and the remote Standby are configured to support TLS connections. That is, the Primary and the remote Standby must either be configured with ACLs or with a private endpoint. See Network Configuration Prerequisites to Allow TLS Authentication for more information.

Autonomous Data Guard Paired Regions

Autonomous Data Guard provides paired regions for cross-region standby databases. A paired region is a region where you can create a remote standby database.

If your tenancy is not subscribed to a paired region for the source region, the region is not displayed in the Region list when you enable Autonomous Data Guard with a cross-region standby or when you add a standby database. To subscribe to a region, see Managing Regions.

If you do not see a region that you are subscribed to in the Region list, open a support ticket to request that the region be added as a paired region for your source region. See Get Help, Search Forums, and Contact Support for more information.
Using Refreshable Clones with Autonomous Database

Autonomous Database provides cloning where you can choose to create a full clone of the active instance, create a metadata clone, or create a refreshable clone. With a refreshable clone the system creates a clone that can be easily updated with changes from the source database.

For details on creating a clone that is a full clone or a metadata clone, see Cloning and Moving an Autonomous Database.

Topics

• About Refreshable Clones on Autonomous Database
• Create a Refreshable Clone for an Autonomous Database Instance
• View Refreshable Clones for an Autonomous Database Instance
• Refresh a Refreshable Clone on Autonomous Database
• Disconnect a Refreshable Clone from the Source Database
• Reconnect a Refreshable Clone to the Source Database
• Use the API
• Refreshable Clone Notes

About Refreshable Clones on Autonomous Database

When you create a refreshable clone for an Autonomous Database instance the system clones the source database to the refreshable clone. After you create a refreshable clone you can refresh the clone with changes from the source database.

Topics

• Refreshable Clone Features
• Refreshable Clone Operations
• Refreshable Clone Lifecycle States
• Refreshable Clone Refresh Timing and Disconnecting from the Source Database
• Refreshable Clone Reconnect to the Source Database
• Operations on an Autonomous Database with an Attached Refreshable Clone

Refreshable Clone Features

A refreshable clone allows you to do the following:
• Maintain one or more copies of the source database for use as read-only databases. A clone database is available when you need it, and when you want to update the data, you can refresh the clone from the source database.

• Share copies of a production database with multiple business units. For example, one business unit might use the source database for ongoing transactions and another business unit could at the same time use the refreshable clone database for read-only operations.

This option also allows you to spread the cost of database usage across multiple business units. You can bill the different units separately, based on their usage of one or more refreshable clone databases.

• Use a refreshable clone as a test database. You can disconnect a refreshable clone from its source and perform DML operations or calculations as needed, in addition to querying data. This allows you to run DML and make changes while the database is disconnected. When you are done with your testing you can reconnect to the source database, which refreshes the clone to the point where it was when you disconnected.

The reconnect operation is only available for 24 hours after the disconnect time. After the reconnect period, the clone is disconnected from the source database and reconnecting is not possible.

Note:
Refreshable clones have a one week refresh age limit. If you do not perform a refresh within a week, then the refreshable clone is no longer refreshable. After a refreshable clone passes the refresh time limit, you can use the instance as a read only database or you can disconnect from the source to make the database a read/write (standard) database.

Refreshable Clone Operations

You can create a refreshable clone from an Autonomous Database instance. After you create a refreshable clone you can perform several operations on the refreshable clone, including: refresh, stop, start, restart, disconnect from source, and terminate.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
</table>
| Create          | You can create a refreshable clone from an Autonomous Database instance. You can create more than one refreshable clone using the same Autonomous Database instance as a source.  
See Create a Refreshable Clone for an Autonomous Database Instance for the steps to create a refreshable clone. |
| View            | You view a refreshable clone from the Oracle Cloud Infrastructure console Autonomous Database Details page.  
See View Refreshable Clones for an Autonomous Database Instance for more information. |
| Start or Restart| When a refreshable clone is stopped as indicated by the Lifecycle State Stopped, you can start the database.  
When a refreshable clone is available as indicated by the Lifecycle State Available, you can restart the database. |
<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh</td>
<td>For a refreshable clone, you can refresh the clone with data from the source database. See Refresh a Refreshable Clone on Autonomous Database for more information.</td>
</tr>
<tr>
<td>Disconnect Clone</td>
<td>You can disconnect a refreshable clone from the source database to make the clone a standard read/write database. See Disconnect a Refreshable Clone from the Source Database for more information.</td>
</tr>
<tr>
<td>Reconnect</td>
<td>When a clone database is disconnected, you can use the clone as a standard read/write database. There is a 24 hour reconnect period where you can reconnect the clone to its source database. After the reconnect period the refreshable clone is disassociated from the source database and reconnecting to the source database is not allowed. See Reconnect a Refreshable Clone to the Source Database for more information.</td>
</tr>
<tr>
<td>Stop</td>
<td>When a refreshable clone is stopped, database operations are not available and charging for OCPU usage on the refreshable clone stops.</td>
</tr>
<tr>
<td>Terminate</td>
<td>If you want to terminate a refreshable clone, select More Actions and Terminate. Terminating a refreshable clone disassociates the clone database from the source database.</td>
</tr>
</tbody>
</table>

### Refreshable Clone Lifecycle States

After you create a refreshable clone, the clone indicates its state on the Autonomous Database Information page in the Lifecycle State field. In addition, the Mode field indicates that a refreshable clone is **Read-Only**.

A refreshable clone indicates its state as follows:

- **Updating**: When a refreshable clone is refreshing or reconnecting, the Lifecycle State field shows **Updating**. While the database is refreshing connections and queries wait until the refresh completes. After the refresh completes the state is set to **Available**, and connections and queries resume.

  See Refresh a Refreshable Clone on Autonomous Database for more information.

- **Stopped**: When a refreshable clone is stopped, database operations are not available and charging for OCPU usage on the refreshable clone stops.

- **Available**: When the refreshable clone is available, database operations are available and you are charged for OCPU usage on the refreshable clone.

The Autonomous Database Information page Mode field indicates the database mode, as follows:

- **Read-Only**: No data can be inserted into or updated in a refreshable clone as it is a read-only database. You can use a refreshable clone for read-only queries and for reporting.

  See Disconnect a Refreshable Clone from the Source Database to change the database to Read/Write mode. In this case the refreshable clone is disconnected from the source database.
Refreshable Clone Refresh Timing and Disconnecting from the Source Database

A banner on the Oracle Cloud Infrastructure console displays the date and time up to which you can refresh the refreshable clone. The banner also includes a Refresh Clone button.

When a refreshable clone is not refreshed within seven (7) days from the last refresh, the banner messages changes to indicate that a refreshable clone that has not been refreshed within seven days cannot be refreshed. The button in the banner changes to Disconnect Clone from Source Database.

When a refreshable clone has not been refreshed within seven days and the refreshable clone has exceeded its maximum refresh time, you have the following options:

- You can continue to use the refreshable clone as a read-only database. The refreshable clone is not refreshable and the data on the refreshable clone reflects the state of the source database at the time of the last successful refresh.
- You can disconnect the refreshable clone from the source database. This disconnects the refreshable clone from the source Autonomous Database instance.

  See Disconnect a Refreshable Clone from the Source Database for more information.

When a refreshable clone has exceeded the maximum refresh time, if you want to use a refreshable clone that can be refreshed from the source database, then you must create a new refreshable clone. If you create a new refreshable clone, then you might also want to terminate the refreshable clone that is no longer able to refresh from the source database.

Refreshable Clone Reconnect to the Source Database

After you perform a disconnect refreshable clone operation, a banner displays the date and time up to which you can reconnect the database to the source. The banner also includes a Reconnect Refreshable Clone button.

When disconnected database is not reconnected within 24 hours from the disconnect time, the Oracle Cloud Infrastructure Console removes the reconnect refreshable clone banner.

When a disconnected refreshable clone has exceeded the reconnect period, you have the following options:

- You can use the database as a standard Autonomous Database; there is no longer an option to reconnect the database to the source database.
Operations on an Autonomous Database with an Attached Refreshable Clone

Describes details for using a source Autonomous Database instance that has one or more attached refreshable clones.

When you make certain changes on a source Autonomous Database instance that has one or more refreshable clones attached to it, the changes are applied to both the source database and to the refreshable clones as follows:

- **Storage:** The storage value you set on the source database applies to both the source database and to any attached refreshable clones.
- **ADMIN password:** The ADMIN password value you set on the source database applies to both the source database and to any attached refreshable clones.

To view the refreshable clones for a source database, on the Autonomous Database Details page, under **Resources**, click **Refreshable Clones**. The Autonomous Database resources area provides a link to each refreshable clone in the Display Name field, and includes the Last Refresh timestamp field and the Refresh Point timestamp field. The refresh point specifies the timestamp for the source database data to which the refreshable clone data is refreshed.

If you want to terminate a source database that has one or more attached refreshable clones, then before you terminate the source database you must do the following until there are no longer any attached refreshable clones. For each attached refreshable clone, do one of the following:

- Disconnect the refreshable clone from the source database. See [Disconnect a Refreshable Clone from the Source Database](#) for more information.
- Terminate the refreshable clone to disassociate the refreshable clone from the source database. You can terminate a refreshable clone by selecting **More Actions** and **Terminate**.

Create a Refreshable Clone for an Autonomous Database Instance

Shows you the steps to create an Autonomous Database refreshable clone from the Oracle Cloud Infrastructure Console.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the [equilateral triangle](#) next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display Name** column.
To create a refreshable clone do the following:

1. On the Details page, from the More Actions drop-down list, select Create Clone.

2. On the Create Autonomous Database Clone page, choose the clone type Refreshable Clone from the choices:
   - **Full Clone**: creates a new database with the source database’s data and metadata.
   - **Refreshable Clone**: To create a refreshable clone, select this clone type.
   - **Metadata Clone**: creates a new database with the source database’s metadata without the data.

3. Provide basic information for the Autonomous Database clone.
   - **Create in Compartment**: See Compartments for information on using and managing compartments.
   - **Source database name**: This field is read-only and shows the name of the source database.
   - **Display name**: Specify a user-friendly description or other information that helps you easily identify the database.
     You can use the name provided, of the form: Clone-of-DBname or change this to the name you want to use to identify the database. The supplied DBname is the name of the source database that you are cloning.
   - **Database name**: Specify the database name; it must consist of letters and numbers only. The maximum length is 30 characters. The same database name cannot be used for multiple Autonomous Databases in the same tenancy in the same region.
     The default database name is a generated 16-character string.

4. Configure the database.
   - **OCPUs count**: Specify the number of CPU cores for your database.
     Your license type determines the OCPU count maximum. For example, if your license type is Bring Your Own License (BYOL) with Oracle Database Standard Edition (SE), the OCPU count maximum is 8.
   - **OCPUs auto scaling**: By default OCPUs auto scaling is enabled to allow the system to automatically use up to three times more CPU and IO resources to meet workload demand. If you do not want to use OCPUs auto scaling then deselect this option.
     See Use Auto Scaling for more information.

**Note:**
The storage for a refreshable clone is set to the same size as on the source database. To change the storage size for a refreshable clone, you must change the storage value on the source database.

5. Choose network access
Note:

After you clone your Autonomous Database you can change the network access option you select for the cloned instance.

- **Secure access from everywhere**
  By default all secure connections are allowed from everywhere.

- **Secure access from allowed IPs and VCNs only**
  This option restricts connections to the database according to the access control lists (ACLs) you specify. To add multiple ACLs for the Autonomous Database, click + Access Control Rule.
  See Configure Access Control Lists When You Provision or Clone an Instance for more information.

- **Private endpoint access only**
  This option assigns a private endpoint, private IP, and hostname to your database. Specifying this option allows traffic only from the VCN you specify; access to the database from all public IPs or VCNs is blocked. This allows you to define security rules, ingress/egress, at the Network Security Group (NSG) level and to control traffic to your Autonomous Database.
  See Configure Private Endpoints When You Provision or Clone an Instance for more information.

6. Choose license and Oracle Database Edition

- **Bring Your Own License (BYOL)**
  Select if your organization already owns Oracle database software licenses. Bring your existing database software licenses to the database cloud service. See [Cloud pricing](#) for information on Bring Your Own License (BYOL) and other licensing options for Oracle Cloud Infrastructure cloud service pricing.

- **Choose an Oracle Database Edition**
  When you select **Bring Your Own License (BYOL)**, you also choose an Oracle Database Edition. The Oracle Database Edition you select is based on the license you bring to Autonomous Database and changes the maximum value that you can select for the **OCPU count**. The choices are:

  **Oracle Database Enterprise Edition (EE):** For this license type the maximum allowed value for **OCPU count** is 128, however you may contact your Oracle account team to request more than 128 OCPUs. With auto scaling enabled you can use up to **OCPU count x 3** OCPUs. For example, if you set the **OCPU count** to 128, you can use up to 384 OCPUs.

  **Oracle Database Standard Edition (SE):** For this license type the maximum allowed value for **OCPU count** is 8. With auto scaling enabled you can use up to **OCPU count x 3** OCPUs. This license restricts the number of OCPUs you can use to a maximum of 8 OCPUs, with or without auto scaling enabled.

  See [Use Auto Scaling](#) for more information.

- **License Included**
  Subscribe to new database software licenses and the database cloud service.

7. (Optional) Provide up to 10 maintenance contacts.
Click **Add Contact** and in the **Contact Email** field, enter a valid email address. If the database you are cloning has a customer contact list, the list is copied. To enter multiple **Contact Email** addresses, repeat the process to add up to 10 customer contact emails.

See [View and Manage Customer Contacts for Operational Issues and Announcements](#) for more information.

8. (Optional) Click **Show Advanced Options** to select advanced options.

   • **Encryption Key**
     Encryption using Oracle-managed keys: By default Autonomous Database uses Oracle-managed encryption keys. Using Oracle-managed keys, Autonomous Database creates and manages the encryption keys that protect your data and Oracle handles rotation of the TDE master key.

     Encrypt using customer-managed keys: If you select customer-managed keys, a master encryption key in the Oracle Cloud Infrastructure Vault is used to generate the TDE master key on Autonomous Database.

     See [Use Customer-Managed Encryption Keys on Autonomous Database](#) for more information.

   • **Maintenance**
     Patch level By default the patch level is the patch level of the source database. Select **Early** to configure the instance with the early patch level.

     When cloning a source database with **Early** patch level, you can only choose the **Early** patch level for your clone.

     See [Set the Patch Level](#) for more information.

   • **Management**
     Shows the character set and national character set for your database.

     See [Choose a Character Set for Autonomous Database](#) for more information.

   • **Tags**
     If you want to use Tags, enter the **TAG KEY** and **VALUE**. Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values which can be attached to resources.

     See [Tagging Overview](#) for more information.

9. Click **Create Autonomous Database Clone**.

On the Oracle Cloud Infrastructure console the **State** shows **Provisioning**... until the refreshable clone is available.

After the provisioning completes the Lifecycle State shows **Available** and the Mode is **Read-only**.

After you download the wallet for the refreshable clone and connect to the database, you can begin using the database to perform read-only operations such as running queries or building reports and notebooks.

After you create a refreshable clone, the Oracle Cloud Infrastructure console shows a banner similar to the following with a message indicating the date prior to which the next refresh must be completed (the banner shows the 7 day refresh limit).
See Refresh a Refreshable Clone on Autonomous Database for details on refreshing a refreshable clone.

**View Refreshable Clones for an Autonomous Database Instance**

If you know a refreshable clone's display name, you can view the instance by selecting from the list of Autonomous Databases on the Oracle Cloud Infrastructure Console. For a refreshable clone, the Display Name field includes a **Refreshable Clone** tag to indicate the instance is a refreshable clone.

The **Clone Information** area shows details for a refreshable clone. While viewing information for a refreshable clone you can view the source database information by clicking the link in the **Source Database** field.
Under Clone Information, the **Refresh Point** value is the timestamp of the source database's data that the clone was last refreshed with.

If an Autonomous Database has attached refreshable clones, you can view the refreshable clones as follows:

1. Choose your region. See **Switching Regions** for information on switching regions and working in multiple regions.
2. Choose your **Compartment**. See **Compartments** for information on using and managing compartments.
3. Select an Autonomous Database instance from the list in your compartment.
4. On the **Autonomous Database Details** page, under **Resources**, click **Refreshable Clones**.

This shows the list of refreshable clones for the Autonomous Database instance.

If you click 🕒 at the end of a row for a refreshable clone in the Refreshable Clones list, you can select an action to perform for the refreshable clone.

### Refresh a Refreshable Clone on Autonomous Database

Shows you the steps to refresh a refreshable clone from the Oracle Cloud Infrastructure Console.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the 📚 next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the **Display Name** column.

You can refresh a refreshable clone as follows:

1. On the **Details** page, from the **More Actions** drop-down list, select **Refresh Clone**.
   
   This shows the Refresh Clone dialog.

2. In the Refresh Clone dialog, specify the **Refresh point timestamp**.
   
   The refresh point specifies the timestamp for the source database data to which the refreshable clone data is refreshed.
The timestamp can be a minimum of 1 minute in the past and a maximum of 7 days in the past. The timestamp must be later than, that is after, the last refresh point. Thus, if you create a refreshable clone, then you would need to wait some time before you refresh the clone.

3. Click Refresh Clone.

On the Oracle Cloud Infrastructure console the Lifecycle State shows Updating until the refresh operation completes. While the database is updating connections and queries wait until the refresh completes.

You can see the status of the refresh operation under Work Requests.

On the Oracle Cloud Infrastructure Console, under Clone Information the Refresh Point shows the timestamp of the source database's data that the clone was refreshed to.

**Note:**

Refreshable clones have a one week refresh age limit. If you do not perform a refresh within a week, then the refreshable clone will not be refreshable. The refresh by date is shown in the Oracle Cloud Infrastructure Console banner. In the case where a refreshable clone is not refreshable, you can disconnect the database from the source to make it a read/write (standard) database. See Disconnect a Refreshable Clone from the Source Database for more information.

**Disconnect a Refreshable Clone from the Source Database**

Shows you the steps to disconnect a refreshable clone from the source database. When you disconnect a refreshable clone the refreshable clone is disassociated from the source database. This converts the database from a refreshable clone to a regular database.
Following the disconnect operation you are allowed to reconnect the disconnected database to the source database. The reconnect operation is limited to a 24 hour reconnect period.

There are several ways to find refreshable clones. See View Refreshable Clones for an Autonomous Database Instance for more information.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click Oracle Database and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the Display Name column.

Starting from the Source Database Console: Disconnect a refreshable clone from the source database and convert the refreshable clone to a read/write database:

1. On the Details page, under Resources select Refreshable Clones.
   This shows the list of refreshable clones.
2. In the row for the refreshable clone you want to disconnect, click the Actions (three vertical dots) icon and select Disconnect Clone from Source Database.
3. In the Disconnect Refreshable Clone dialog enter the source database name to confirm disconnecting the clone.
4. Click Disconnect Refreshable Clone.

Starting from the Clone Database Console: Disconnect a refreshable clone from the source database and convert the refreshable clone to a read/write database:

1. On the Details page, under Clone Information click Disconnect.
2. In the Disconnect Refreshable Clone dialog enter the source database name to confirm disconnecting the clone.
3. Click Disconnect Refreshable Clone.

The Autonomous Database Lifecycle State of the cloned database changes to Updating. When the disconnect operation completes the Lifecycle State changes to Available and the Mode shows Read/Write.

After you disconnect a refreshable clone, the Oracle Cloud Infrastructure Console updates with the following changes:

- The Display Name on the Autonomous Databases page updates and does not show the Refreshable Clone indicator for the cloned instance.
- On the disconnected instance, the Autonomous Database Details page updates and does not show the Refreshable Clone indicator, and the Clone Information area is removed.
- On the disconnected instance, the Autonomous Database details page displays a banner indicating the date and time up to which you can reconnect the database to the source. The banner also includes a Reconnect Refreshable Clone button.
For example:

![Image of reconnect refreshable clone dialog]

See [Reconnect a Refreshable Clone to the Source Database](#) for more information.

- On the source database, on the [Autonomous Database Details](#) page, when you click [Refreshable Clones](#) under [Resources](#), the list no longer shows an entry for the disconnected clone.

**Notes for disconnecting a refreshable clone.**

- Disconnecting a refreshable clone from the source may fail if the source database has scaled down its storage and the refreshable clone has a larger amount of data than the source. In this case, you have the following options:
  - You may scale up the storage on the source database temporarily before disconnecting, and then scale the source back down after disconnecting.
  - Refresh the clone to a point where the scale down of storage occurred.

- After you disconnect a refreshable clone you have 24 hours to reconnect. After the reconnect period is over, the reconnect operation is not available. If you do not reconnect a disconnected refreshable clone, the clone is a standard Autonomous Database and there is no longer an option to reconnect the database to the source database.

- A disconnected refreshable clone is no longer associated with the source database. To use the database or to initiate the reconnect operation, you must know the name of the refreshable clone database that was disconnected from the source database. You must initiate the reconnect operation from the disconnected clone database. You cannot reconnect a disconnected clone from the source database.

---

### Reconnect a Refreshable Clone to the Source Database

Shows you the steps to reconnect a disconnected clone to the source database. After you disconnect a refreshable clone, during the following 24 hour reconnect period you can reconnect to the source database. The reconnect operation restores all the data back to that time when the clone was disconnected.

Perform the following prerequisite steps as necessary:

- Open the Oracle Cloud Infrastructure Console by clicking the [≡](#) next to Oracle Cloud.
- From the Oracle Cloud Infrastructure left navigation menu click [Oracle Database](#) and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.
- On the Autonomous Databases page select your Autonomous Database from the links under the [Display Name](#) column.

Reconnect a database and restore all the data back to that time when the refreshable clone was disconnected from the source database as follows:

1. On the [Details](#) page, from the [More Actions](#) drop-down list, select [Reconnect Refreshable Clone](#).

   This shows the Reconnect Refreshable Clone dialog.

2. In the Reconnect Refreshable Clone dialog, enter the source database name to confirm.
Note:

All data and metadata inserted, updated, or deleted from the database where you run the reconnect operation will be lost when the database is reconnected to the source database.

3. Click **Reconnect Refreshable Clone**.

The Autonomous Database Lifecycle State changes to **Updating**.

When the reconnect refreshable clone operation completes, the Oracle Cloud Infrastructure Console on the clone indicates the change as follows:

- The Mode shows **Read Only**.
- If the clone was available when you initiated the reconnect operation, the Lifecycle State changes to **Available**.
- If the clone was stopped when you initiated the reconnect operation, the Lifecycle State shows **Stopped**.
- On the Autonomous Database Details page, the Clone Information area is updated. The **Clone Type** shows Refreshable Clone, the **Source Database** shows the source database link, the refresh (Refresh Clone) link, and the disconnect link (Disconnect Refreshable Clone), and the **Refresh Point** shows the last refresh point timestamp.

Notes for reconnecting a disconnected clone to the source database:

- If you disconnect a refreshable clone, you have 24 hours to use the reconnect operation. After the reconnect period is over, the reconnect operation is not available. If you do not reconnect a disconnected refreshable clone, the Autonomous Database is a standard read/write database and there is no longer an option to reconnect the database to the source database.
- Network ACLs and Private Endpoint network configuration options on a refreshable clone can be changed when the clone is disconnected from the source database. When you reconnect a refreshable clone to its source database, the reconnect operation does not restore the network ACLs or Private Endpoint network configuration options on the refreshable clone.

**Use the API**

For information about using the API and signing requests, see [REST APIs and Security Credentials](#). For information about SDKs, see [Software Development Kits and Command Line Interface](#).

Use these API operations to manage a refreshable clone:

- To create a refreshable clone, use `CreateAutonomousDatabase`
- To refresh a refreshable clone, use `AutonomousDatabaseManualRefresh`
- To list the attached refreshable clones for a database, use `ListAutonomousDatabaseClones`
- To disconnect a refreshable clone from the source database, use `UpdateAutonomousDatabase`
Refreshable Clone Notes

Lists limitations and notes for Autonomous Database refreshable clones.

- Always Free Autonomous Databases do not support refreshable clones.
- Autonomous Database does not support using customer-managed encryption keys with a refreshable clone. You cannot create a refreshable clone from a source database that uses customer-managed encryption keys. Additionally, you cannot switch to a customer-managed encryption key on a source database that has one or more refreshable clones.
- You cannot create a cascading series of refreshable clones. Thus, a refreshable clone cannot be created from another refreshable clone.
- You cannot backup or restore a refreshable clone.
- The ADMIN password for a refreshable clone is inherited from the source database. If you want to change the ADMIN password for a refreshable clone you must change the ADMIN password on the source database, and then refresh the clone for the ADMIN password on the clone to come into effect.
- Oracle APEX (APEX) URLs do not work in a refreshable clone read-only database and the APEX URLs are disabled in the Oracle Cloud Infrastructure Console and in the Database Actions APEX and APEX Workspaces cards. APEX URLs are enabled for a read/write database when a refreshable clone is disconnected from the source.
- Oracle Machine Learning is disabled in a refreshable clone read-only database. The OML User Administration URLs are disabled in the Oracle Cloud Infrastructure Console and on the Database Actions Launchpad.
- Oracle Data Safe is not supported for a refreshable clone instance. The Data Safe data in the refreshable clone's source database, for example user registration, data masking, and so on are available and Data Safe can be enabled when a refreshable clone is disconnected from the source.
- When you scale up or scale down the storage on the source Autonomous Database instance for a refreshable clone, the change is immediately reflected in the database console and in the billing for the refreshable clone. When the refreshable clone is refreshed to a refresh point after the scale up or down operation, the system makes a corresponding change to the refreshable clone storage (scaling up or scaling down the refreshable clone storage to match the source database).
- You cannot use the rename operation on a refreshable clone instance or on a database that is the source for a refreshable clone.
- You cannot set the patch level when you create a refreshable clone. A refreshable clone has the same patch level as the source database. See Set the Patch Level for more information.
- Automatic Workload Repository (AWR) data and reports are not available for refreshable clones. In addition, the graphs that rely on AWR data are not available, including the following graphs:
  - The Running SQL statements graph on the Overview tab shown on the Database Monitor card in Database Actions.
  - The SQL statement response time graph on the Overview tab shown on the Database Monitor card in Database Actions.
  - The Time period graphs on the Monitor tab shown on the Database Monitor card in Database Actions.
- The Performance Hub graph data older than one hour is not available.

- For an Oracle Autonomous JSON Database (workload type JSON Database), note the following when reconnecting to the source database:
  - If, after you disconnect the refreshable clone, you promote both the clone and the source to Oracle Autonomous Transaction Processing (workload type Transaction Processing), you can reconnect the database to the source.
  - If after you disconnect the refreshable clone, you promote the source database to Oracle Autonomous Transaction Processing (workload type Transaction Processing) and do not promote the disconnected clone, the disconnected clone must be promoted to before you perform the reconnect operation.
  - If after you disconnect the refreshable clone, you promote the disconnected database to Oracle Autonomous Transaction Processing (workload type Transaction Processing), you can still reconnect to the source but the reconnected database remains in the promoted state.

See Promote to Autonomous Transaction Processing for more information.
Auditing Autonomous Database

Autonomous Database provides auditing that allows you to monitor Oracle database activities.

Topics

- About Auditing Autonomous Database
- Enable and Register Oracle Data Safe on Autonomous Database
- Extend Audit Record Retention with Oracle Data Safe on Autonomous Database
- View and Manage Oracle Data Safe Audit Trails on Autonomous Database
- View and Manage Audit Policies with Oracle Data Safe on Autonomous Database
- Generate Audit Reports with Data Safe on Autonomous Database

About Auditing Autonomous Database

Autonomous Database provides auditing to track, monitor, and record database actions. Auditing can help you detect security risks and improve regulatory compliance for your database.

Topics

- Audit Features on Autonomous Database
- Audit Data on Autonomous Database
- Default Audit Policies on Autonomous Database

Audit Features on Autonomous Database

Autonomous Database includes extensive, sophisticated audit capabilities that allow you capture the audit information you need for your organization. Autonomous Database provides default auditing.

In addition, you can use either of the following to apply auditing policies:

- Use Oracle Data Safe to apply auditing policies for database users, for administrative users, and to apply predefined auditing policies or to apply customized auditing policies. See Activity Auditing Overview for more information.
- Configure Oracle Database Audit Policies. See Configuring Audit Policies for more information.

You can configure auditing to accomplish the following:

- Enable accountability for actions. These include actions taken in a particular schema, table, or row, or affecting specific content.
• Deter users, or others, such as intruders, from inappropriate actions based on their accountability.

• Investigate suspicious activity. For example, if a user is logging into the database using the application's database credentials, then auditing connections to the database lets you determine that the login came from a user's workstation instead of from the application server.

• Notify an auditor of the actions of an unauthorized user. For example, notify an auditor when an unauthorized user attempts to delete data from a table.

• Monitor and gather data about specific database activities. For example, you can gather statistics about which tables are being updated, the number of failed logins, or how many concurrent users connect at peak times.

• Detect problems with an authorization or access control implementation. For example, you can create audit policies that you expect will never generate an audit record because the data is protected in other ways. However, if these policies generate audit records, then you will know the other security controls are not properly implemented.

• Address auditing requirements for compliance. Regulations such as the following have common auditing-related requirements:
  – European Union General Data Protection Regulation (GDPR)
  – Sarbanes-Oxley Act
  – Health Insurance Portability and Accountability Act (HIPAA)
  – Japan Privacy Law
  – European Union Directive on Privacy and Electronic Communications

Audit Data on Autonomous Database

Autonomous Database protects audit data and writes its audit trail to the UNIFIED_AUDIT_TRAIL data dictionary view.

The underlying table storing audit data on Autonomous Database is AUDSYS.AUD$UNIFIED. This table is protected and does not allow users to perform DML/DDL operations or to purge the table (any attempt to perform these actions automatically produces an audit record). After an audit record is written, the only activity allowed is for the ADMIN user to perform a PURGE. The ADMIN has the AUDIT_ADMIN role that is required to run a PURGE. If you assign the AUDIT_ADMIN role to another user, then that user could also perform a PURGE.

Depending on the number and type of audit policies you use and the amount of activity, over time the audit trail can grow to use a large amount of storage. Autonomous Database provides the following ways to limit the storage required for audit data:

• Each Autonomous Database instance runs an automated purge job once a day to remove all audit records older than fourteen (14) days.

• Users with the AUDIT_ADMIN role can purge audit records manually using the DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL procedure. See DBMS_AUDIT_MGMT for more information.
If you need a longer audit data retention period than 14 days, use Oracle Data Safe to retain audit data. See Extend Audit Record Retention with Oracle Data Safe on Autonomous Database for more information.

Default Audit Policies on Autonomous Database

Autonomous Database provides auditing to track, monitor, and record activities on your database.

By default, Autonomous Database applies audit policies to audit the following database activities:

- All activity by Oracle Cloud Operations
- All login failures to the database
- All password changes
- Attempts to create or alter procedures
- Execution of certain procedures, including procedures in the packages: UTL_HTTP or UTL_SMTP that connect to the network

In addition, you can use either of the following to apply additional auditing policies:

- Use Oracle Data Safe to apply auditing policies for database users, for administrative users, and to apply predefined auditing policies or to apply customized auditing policies. For more information, see:
  - View and Manage Audit Trails
  - View and Manage Audit Policies
- Configure Oracle Database Audit Policies. See Configuring Audit Policies for more information.

Enable and Register Oracle Data Safe on Autonomous Database

Use Oracle Data Safe to apply auditing policies for database users, for administrative users, to apply predefined auditing policies or to extend the audit data record retention for your Autonomous Database instance.

Note:

Oracle Data Safe is not available with Always Free Autonomous Database.

Enable Oracle Data Safe and register your Autonomous Database instance with Oracle Data Safe as follows:

1. Enable Oracle Data Safe. See Enable Oracle Data Safe for more information.
2. Register your Autonomous Database instance with Oracle Data Safe.
   a. Open the Oracle Cloud Infrastructure Console by clicking the next to Oracle Cloud.
b. From the Oracle Cloud Infrastructure left navigation menu click **Oracle Database** and then, depending on your workload click one of: Autonomous Data Warehouse, Autonomous JSON Database, or Autonomous Transaction Processing.

c. On the Autonomous Databases page select an Autonomous Database from the links under the **Display Name** column.

d. On the Autonomous Database Details page, under **Data Safe**, click **Register**.

e. In the Register Database with Data Safe dialog, click **Confirm**.

The Data Safe status shows: **Registering**. This step takes about 15 minutes.

After you are registered, the Data Safe status shows **Registered** and there are two links: **View** and **Deregister**.

---

**Extend Audit Record Retention with Oracle Data Safe on Autonomous Database**

Use Oracle Data Safe to extend the audit data record retention to a specified number of months.

---

**Note:**

Oracle Data Safe is not available with Always Free Autonomous Database.

First enable Oracle Data Safe and register your Autonomous Database instance with Oracle Data Safe. See **Enable and Register Oracle Data Safe on Autonomous Database** for more information.
After Oracle Data Safe is enabled and your Autonomous Database instance is registered, you can specify the Data Safe retention period.

See Update Retention Periods for a Target Database for more information.

View and Manage Oracle Data Safe Audit Trails on Autonomous Database

Data Safe uses audit trails to define where to retrieve the audit data and to collect Autonomous Database audit records. During the registration process, Oracle Data Safe discovers the audit trails and creates an audit trail resource.

Oracle Data Safe lists resources on the Audit Trails page in Security Center. You can discover new audit trails at any time and remove audit trail resources in Oracle Data Safe as needed.

First enable Oracle Data Safe and register your Autonomous Database instance with Oracle Data Safe. See Enable and Register Oracle Data Safe on Autonomous Database for more information.

See View and Manage Audit Trails for more information.

Note:

When the Autonomous Database is stopped or restarted, the audit trail switches to a STOPPED state. Data Safe makes multiple attempts to reconnect for fifteen (15) minutes. If Data Safe cannot reconnect, you can manually start the audit trail.

You can also manually stop or delete the audit trail. Deleting the audit trail does not remove audit records that have already been collected. Those records remain in Data Safe until the retention period is reached.

View and Manage Audit Policies with Oracle Data Safe on Autonomous Database

Use Oracle Data Safe to set audit policies for your Autonomous Database instance.

Note:

Oracle Data Safe is not available with Always Free Autonomous Database.

First enable Oracle Data Safe and register your Autonomous Database instance with Oracle Data Safe. See Enable and Register Oracle Data Safe on Autonomous Database for more information.

After Oracle Data Safe is enabled and your Autonomous Database instance is registered, access Oracle Data Safe to set audit policies.

See View and Manage Audit Policies for more information.
Generate Audit Reports with Data Safe on Autonomous Database

Data Safe includes out-of-box audit data reports, and you can create custom reports to suit your needs.

After you enable and register Oracle Data Safe, and you add a trail to collect audit data from your Autonomous Database instance, then you can use the reports to monitor activity for your database.

See View and Manage Audit Reports for more information.
Part IV
Appendixes

Part III contains the Appendixes.

Topics

- Autonomous Database Supplied Package Reference
- Autonomous Database for Experienced Oracle Database Users
- Migrating MySQL and Third-Party Databases to Autonomous Database
- Sample Star Schema Benchmark (SSB) Queries and Analytic Views
- SODA Collection Metadata on Autonomous Database
- Obtain Tenancy Details
Autonomous Database Supplied Package Reference

This appendix provides information about the packages you use with Autonomous Database. The DBMS_CLOUD topic also covers the DBMS_CLOUD REST APIs.

Topics

- DBMS_CLOUD Package
- DBMS_CLOUD_ADMIN Package
- DBMS_CLOUD_MACADM Package
- DBMS_CLOUD_REPO Package
- DBMS_DCAT Package
- DBMS_MAX_STRING_SIZE Package
- DBMS_AUTO_PARTITION Package
- CS_RESOURCE_MANAGER Package
- CS_SESSION Package

DBMS_CLOUD Package

The DBMS_CLOUD package provides database routines for working with cloud resources.

Topics

- DBMS_CLOUD Subprograms and REST APIs
- DBMS_CLOUD Package File URI Formats
- DBMS_CLOUD Package Format Options
- DBMS_CLOUD Package Format Options for EXPORT_DATA with Text Files (CSV, JSON, and XML)
- DBMS_CLOUD Avro, ORC, and Parquet Support
- DBMS_CLOUD Exceptions

DBMS_CLOUD Subprograms and REST APIs

This section covers the DBMS_CLOUD subprograms and REST APIs provided with Autonomous Database.
Note:

To run `DBMS_CLOUD` subprograms with a user other than ADMIN you need to grant `EXECUTE` privileges to that user. For example, run the following command as ADMIN to grant privileges to `adb_user`:

```
GRANT EXECUTE ON DBMS_CLOUD TO adb_user;
```

This `DBMS_CLOUD` package is made up of the following:

• `DBMS_CLOUD` for Access Management
• `DBMS_CLOUD` for Objects and Files
• `DBMS_CLOUD` REST APIs

**DBMS_CLOUD for Access Management**

The subprograms for credential management within the `DBMS_CLOUD` package, including creating, deleting, and updating credentials.

<table>
<thead>
<tr>
<th>Subprogram</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CREATE_CREDENTIAL Procedure</code></td>
<td>This procedure stores cloud service credentials in Autonomous Database.</td>
</tr>
<tr>
<td><code>DROP_CREDENTIAL Procedure</code></td>
<td>This procedure removes an existing credential from Autonomous Database.</td>
</tr>
<tr>
<td><code>UPDATE_CREDENTIAL Procedure</code></td>
<td>This procedure updates cloud service credential attributes in Autonomous Database.</td>
</tr>
</tbody>
</table>

**CREATE_CREDENTIAL Procedure**

This procedure stores cloud service credentials in Autonomous Database.

Use stored cloud service credentials to access the cloud service for data loading, for querying external data residing in the cloud, or for other cases when you use `DBMS_CLOUD` procedures with a `credential_name` parameter. This procedure is overloaded:

• Use the Oracle Cloud Infrastructure-related parameters, including: `user_ocid`, `tenancy_ocid`, `private_key`, and `fingerprint` only when you are using Oracle Cloud Infrastructure Signing Keys authentication.

• Use the AWS ARN-related parameter `params`, only when you are using Amazon Resource Names (ARNs) credentials.

**Syntax**

```sql
DBMS_CLOUD.CREATE_CREDENTIAL (credential_name  IN VARCHAR2,
username       IN VARCHAR2,
password       IN VARCHAR2 DEFAULT NULL);
```
DBMS_CLOUD.CREATE_CREDENTIAL (  
    credential_name IN VARCHAR2,
    user_ocid IN VARCHAR2,
    tenancy_ocid IN VARCHAR2,
    private_key IN VARCHAR2,
    fingerprint IN VARCHAR2);

DBMS_CLOUD.CREATE_CREDENTIAL (  
    credential_name IN VARCHAR2,
    params   IN CLOB DEFAULT);

Parameters

<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. The credential_name parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.</td>
</tr>
<tr>
<td>username</td>
<td>The username and password arguments together specify your cloud service credentials. See the usage notes for what to specify for the username and password for different cloud services.</td>
</tr>
<tr>
<td>password</td>
<td>The username and password arguments together specify your cloud service credentials.</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>
<tr>
<td>params</td>
<td>Specifies credential parameters using Amazon Resource Names (ARNs) 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.
Oracle Cloud Infrastructure Credentials (Auth Tokens)

For Oracle Cloud Infrastructure the username is your Oracle Cloud Infrastructure user name. The password is your Oracle Cloud Infrastructure auth token. See Working with Auth Tokens.

For example:

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

Oracle Cloud Infrastructure Signing Key Based Credentials

Use the Oracle Cloud Infrastructure signing key related parameters, including: user_ocid, tenancy_ocid, private_key, and fingerprint with Oracle Cloud Infrastructure Signing Keys authentication.

For example:

BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'OCI_KEY_CRED',
        user_ocid => 'ocid1.user.oc1..aaaaaaaauq54mi7zdyfhw33ozkuontjcee17fok5nq3bf2wetkpqsoa',
        tenancy_ocid => 'ocid1.tenancy.oc1..aabbbbbbaafcue47pqmrfr4vigneobgcmmy5r7xvoypicjqqqge32ewnracyx2a',
        private_key => 'MIIEogIBAAKCAQEAtUnxbmrekwgVac6FwDeRzoXvIpA9+0r1.....wtmNpESQQQQLGPD8
NM//JEBg=',
END;
/

Private keys generated with a passphrase are not supported. You need to generate the private key without a passphrase. See How to Generate an API Signing Key for more information.

Oracle Cloud Infrastructure Object Storage Classic Credentials

If your source files reside in Oracle Cloud Infrastructure Object Storage Classic, the username is your Oracle Cloud Infrastructure Classic user name and the password is your Oracle Cloud Infrastructure Classic password.
Amazon Web Services (AWS) Credentials

If your source files reside in Amazon S3 or you are calling an AWS API, the username is your AWS access key ID and the password is your AWS secret access key. See AWS Identity and Access Management.

Microsoft Azure Credentials

If your source files reside in Azure Blob Storage or you are calling an Azure API, the username is your Azure storage account name and the password is an Azure storage account access key. See About Azure storage accounts.

Amazon S3-Compatible Credentials

<table>
<thead>
<tr>
<th>Service</th>
<th>Credentials Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Cloud Infrastructure (Customer Secret Keys)</td>
<td>If your source files reside in Oracle Cloud Infrastructure, then you need to use Customer Secret Keys with S3-compatible URLs. See Working with Customer Secret Keys for more information.</td>
</tr>
<tr>
<td>Google Cloud Storage</td>
<td>If your source files reside in Google Cloud Storage or you are calling Google Cloud Storage APIs, then you need to set a default Google project and obtain an HMAC key to create credentials to supply with Google Cloud Storage S3-compatible URLs. Use the HMAC key id as the username, and the HMAC secret as the password. See Projects and HMAC Keys for more information.</td>
</tr>
<tr>
<td>Wasabi Hot Cloud Storage</td>
<td>If your source files reside in Wasabi Hot Cloud Storage or you are calling Wasabi Hot Cloud Storage APIs, then you need Access Keys to create credentials to supply with S3-compatible URLs. Use the Wasabi Hot Cloud Storage Access Key as the username, and the Wasabi Hot Cloud Storage Secret Key as the password. See Creating a Wasabi API Access Key Set for more information.</td>
</tr>
</tbody>
</table>

AWS Amazon Resource Names (ARN) Credentials

If your source files reside in Amazon S3 or you are calling an AWS API, use params to specify the parameters for the Amazon Resource Names (ARN).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws_role_arn</td>
<td>Specifies the Amazon Resource Name (ARN) that identifies the AWS role. If this parameter is not supplied when creating the credential, ORA-20041 is raised.</td>
</tr>
<tr>
<td>external_id_type</td>
<td>Optionally set the external_id_type to use the Autonomous Database compartment OCID, database OCID, or tenancy OCID by supplying one of: compartment_ocid, database_ocid, or tenant_ocid. If this parameter is not given when creating the credential, the default value is database_ocid.</td>
</tr>
</tbody>
</table>

For example:

```
BEGIN DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name       => 'MY_CRED',
```
GitHub Personal Access Token

If your source files reside in a GitHub repository or you are calling a GitHub API, the username is your GitHub email and the password is your GitHub personal access token. See Creating a personal access token for more information.

For example:

BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'MY_GITHUB_CRED',
        username => 'user@example.com',
        password => 'your_personal_access_token');
END;
/

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>

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.

Syntax

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></td>
<td>For a credential for an Amazon ARN, the valid attribute values are:</td>
</tr>
<tr>
<td></td>
<td>aws_role_arn and external_id_type.</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.

Examples

```
BEGIN
    DBMS_CLOUD.UPDATE_CREDENTIAL(
        credential_name => 'OBJ_STORE_CRED',
        attribute => 'PASSWORD',
        value => 'password');
END;
/
```

```
BEGIN
    DBMS_CLOUD.UPDATE_CREDENTIAL(
        credential_name => 'ARN_CRED',
        attribute => 'aws_role_arn',
        value => 'NEW_AWS_ARN');
END;
/
```

DBMS_CLOUD for Objects and Files

The subprograms for object and file management within the DBMS_CLOUD package.

<table>
<thead>
<tr>
<th>Subprogram</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPY_COLLECTION Procedure</td>
<td>This procedure loads data into existing SODA collection from Cloud Object Storage.</td>
</tr>
<tr>
<td>COPY_DATA Procedure</td>
<td>This procedure loads data into existing Autonomous Database tables from files in the Cloud.</td>
</tr>
<tr>
<td>Subprogram</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>COPY_DATA Procedure for Avro, ORC, or Parquet Files</td>
<td>This procedure with the <code>format</code> parameter type set to the value <code>orc</code>, <code>parquet</code>, or <code>avro</code> loads data into existing Autonomous Database tables from ORC, Parquet, or Avro files in the Cloud. Similar to text files, the data is copied from the source ORC, Parquet, or Avro file into the preexisting internal table.</td>
</tr>
<tr>
<td>CREATE_EXTERNAL_TABLE Procedure</td>
<td>This procedure creates an external table on files in the Cloud. This allows you to run queries on external data from Autonomous Database.</td>
</tr>
<tr>
<td>CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files</td>
<td>This procedure with the <code>format</code> parameter type set to the value <code>parquet</code>, <code>orc</code>, or <code>avro</code>, creates an external table with either Parquet, ORC, or Avro format files in the Cloud. This allows you to run queries on external data from Autonomous Database.</td>
</tr>
<tr>
<td>CREATE_EXTERNAL_PART_TABLE Procedure</td>
<td>This procedure creates an external partitioned table on files in the Cloud. This allows you to run queries on external data from Autonomous Database.</td>
</tr>
<tr>
<td>CREATE_HYBRID_PART_TABLE Procedure</td>
<td>This procedure creates a hybrid partitioned table. This allows you to run queries on hybrid partitioned data from Autonomous Database.</td>
</tr>
<tr>
<td>DELETE_ALL_OPERATIONS Procedure</td>
<td>This procedure clears either all data load operations logged in the <code>user_load_operations</code> table in your schema or clears all the data load operations of the specified type, as indicated with the <code>type</code> parameter.</td>
</tr>
<tr>
<td>DELETE_FILE Procedure</td>
<td>This procedure removes the specified file from the specified directory on Autonomous Database.</td>
</tr>
<tr>
<td>DELETE_OBJECT Procedure</td>
<td>This procedure deletes the specified object on object store.</td>
</tr>
<tr>
<td>EXPORT_DATA Procedure</td>
<td>This procedure exports data from Autonomous Database to files in the Cloud based on the result of the specified SQL query. The overloaded form enables you to use the <code>operation_id</code> parameter. Depending on the <code>format</code> parameter specified, the procedure exports rows to the Cloud Object store as text format with options of CSV, JSON, or XML format files or using the ORACLE_DATAPUMP access driver to write data to a dump file.</td>
</tr>
<tr>
<td>GET_OBJECT Procedure and Function</td>
<td>This procedure is overloaded. The procedure form reads an object from Cloud Object Storage and copies it to Autonomous Database. The function form reads an object from Cloud Object Storage and returns a BLOB to Autonomous Database.</td>
</tr>
<tr>
<td>LIST_FILES Function</td>
<td>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.</td>
</tr>
<tr>
<td>LIST_OBJECTS Function</td>
<td>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.</td>
</tr>
<tr>
<td>Subprogram</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PUT_OBJECT Procedure</td>
<td>This procedure is overloaded. In one form the procedure copies a file from Autonomous Database to the Cloud Object Storage. In another form the procedure copies a BLOB from Autonomous Database to the Cloud Object Storage.</td>
</tr>
<tr>
<td>SYNC_EXTERNAL_PART_TABLE Procedure</td>
<td>This procedure simplifies updating an external partitioned table from files in the Cloud. Run this procedure whenever new partitions are added or when partitions are removed from the Object Store source for the external partitioned table.</td>
</tr>
<tr>
<td>VALIDATE_EXTERNAL_TABLE Procedure</td>
<td>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.</td>
</tr>
<tr>
<td>VALIDATE_EXTERNAL_PART_TABLE Procedure</td>
<td>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.</td>
</tr>
<tr>
<td>VALIDATE_HYBRID_PART_TABLE Procedure</td>
<td>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.</td>
</tr>
</tbody>
</table>

**COPY_COLLECTION Procedure**

This procedure loads data into a SODA collection from Cloud Object Storage. If the specified SODA collection does not exist, the procedure creates it. The overloaded form enables you to use the operation_id parameter.

**Syntax**

```sql
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. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL Procedure for more information.</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. See DBMS_CLOUD Package File URI Formats for more information.</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: characterset, compression, ignoreblanklines, jsonpath, maxdcoze, recorddelimiter, rejectlimit, type, unpackarrays. Apart from the mentioned formats for JSON data, Autonomous Database supports other formats too. For the list of format arguments supported by Autonomous Database, see DBMS_CLOUD Package Format Options.</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/adbexample/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. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL Procedure for more information.</td>
</tr>
<tr>
<td>file_uri_list</td>
<td>This parameter specifies a 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. When the format parameter type option value is json, this parameter is ignored. For an example using field_list, see CREATE_EXTERNAL_TABLE Procedure.</td>
</tr>
</tbody>
</table>
Parameter | Description
---|---
format | The options describing the format of the source files. For the list of the options and how to specify the values see DBMS_CLOUD Package Format Options. For Avro, ORC, or Parquet file format options, see DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet.
operation_id | Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Usage Note

The default record delimiter is detected newline. With detected newline, DBMS_CLOUD tries to automatically find the correct newline character to use as the record delimiter. DBMS_CLOUD first searches for the Windows newline character \r\n. If it finds the Windows newline character, this is used as the record delimiter for all files in the procedure. If a Windows newline character is not found, DBMS_CLOUD searches for the UNIX/Linux newline character \n, and if it finds one it uses \n as the record delimiter for all files in the procedure. If the source files use a combination of different record delimiters, you may encounter an error such as, "KUP-04020: found record longer than buffer size supported". In this case, you need to either modify the source files to use the same record delimiter or only specify the source files that use the same record delimiter.

See DBMS_CLOUD Package Format Options for information on the recorddelimier format option.

COPY_DATA Procedure for Avro, ORC, or Parquet Files

This procedure with the format parameter type set to the value avro, orc, or parquet loads data into existing Autonomous Database tables from Avro, ORC, or Parquet files in the Cloud. Similar to text files, the data is copied from the source Avro, ORC, or Parquet file into the preexisting internal table.

Syntax

```
DBMS_CLOUD.COPY_DATA (  
    table_name        IN VARCHAR2,  
    credential_name   IN VARCHAR2,  
    file_uri_list     IN CLOB,  
    schema_name       IN VARCHAR2 DEFAULT,  
    field_list        IN CLOB DEFAULT,  
    format            IN CLOB DEFAULT);  
```

Parameters

Parameter | Description
---|---
table_name | The name of the target table on the database. The target table needs to be created before you run COPY_DATA.
<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. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL Procedure for more information.</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 Avro, ORC, or Parquet files. The fields in the source match the external table columns by name. Source data types are converted to the external table column data type. For ORC files, see DBMS_CLOUD Package ORC to Oracle Data Type Mapping For Parquet files, see DBMS_CLOUD Package Parquet to Oracle Data Type Mapping for details on mapping. For Avro files, see DBMS_CLOUD Package Avro to Oracle Data Type Mapping for details on mapping.</td>
</tr>
<tr>
<td>format</td>
<td>The options describing the format of the source files. For Avro, ORC, or Parquet files, only two options are supported: see DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet.</td>
</tr>
</tbody>
</table>

Usage Notes

- As with other data files, Avro, ORC, and Parquet data loads generate logs that are viewable in the tables dba_load_operations and user_load_operations. Each load operation adds a record to dba[<user>].load_operations that indicates the table containing the logs. The log table provides summary information about the load.

- For Avro, ORC, or Parquet, when the format parameter type is set to the value avro, orc, or parquet, the BADFILE_TABLE table is always empty.
  - For Parquet files, PRIMARY KEY constraint errors throw an ORA error.
  - If data for a column encounters a conversion error, for example, the target column is not large enough to hold the converted value, the value for the column is set to NULL. This does not produce a rejected record.

CREATE_EXTERNAL_PART_TABLE Procedure

This procedure creates an external partitioned table on files in the Cloud. This allows you to run queries on external data from Autonomous Database.
Syntax

DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE (  
    table_name           IN VARCHAR2,  
    credential_name      IN VARCHAR2,  
    partitioning_clause  IN CLOB,  
    column_list          IN CLOB,  
    field_list           IN CLOB DEFAULT,  
    format               IN CLOB DEFAULT);  

DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE (  
    table_name           IN VARCHAR2,  
    credential_name      IN VARCHAR2,  
    file_uri_list        IN VARCHAR2,  
    column_list          IN CLOB,  
    field_list           IN CLOB DEFAULT,  
    format               IN CLOB DEFAULT);  

Parameters

<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. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL Procedure for more information.</td>
</tr>
<tr>
<td>partitioning_clause</td>
<td>Specifies the complete partitioning clause, including the location information for individual partitions. If you use the partitioning_clause parameter, the file_uri_list parameter is not allowed.</td>
</tr>
<tr>
<td>file_uri_list</td>
<td>There are two options for the file_uri_list parameter: A common-delimited list of individual file URIs without wildcards. A single file URI with wildcards, and the wildcards can only be after the last slash &quot;/&quot;. If you use the parameter file_uri_list, the partitioning_clause parameter is not allowed. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD Package File URI Formats for more information.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>column_list</td>
<td>Comma-delimited list of column names and data types for the external table. This parameter has the following requirements, depending on the type of the data files specified with the file_url_list parameter:</td>
</tr>
<tr>
<td></td>
<td>• The column_list parameter is required with unstructured files. Using unstructured files, for example with CSV text files, the column_list parameter must specify all the column names and data types inside the data file as well as the partition columns derived from the object name.</td>
</tr>
<tr>
<td></td>
<td>• The column_list parameter is optional with structured files. For example, with Avro, ORC, or Parquet data files, the column_list is not required. When the column_list is not included, the format parameter partition_columns option must include specifications for both column names (name) and data types (type).</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>
</tbody>
</table>
The format option `partition_columns` specifies the `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` column names and data types of partition columns when the partition columns are derived from the file path, depending on the type of data file, structured or unstructured:

- **When the `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` includes the `column_list` parameter and the data files are unstructured, such as with CSV text files, `partition_columns` does not include the data type. For example, use a format such as the following for this type of `partition_columns` specification:**

  ```json
  "partition_columns": ["state","zipcode"]
  ```

  The data type is not required because it is specified in the `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE column_list parameter`.

- **When the `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` does not include the `column_list` parameter and the data files are structured, such as Avro, ORC, or Parquet files, the `partition_columns` option includes both the column name, name sub-clause, and the data type, type sub-clause. For example, the following shows a `partition_columns` specification:**

  ```json
  "partition_columns": [
    {"name":"country","type":"varchar2(10)",
    {"name":"year","type":"number"},
    {"name":"month","type":"varchar2(10)"}
  ]
  ```

  If the data files are unstructured and the type sub-clause is specified with `partition_columns`, the type sub-clause is ignored.

  For object names that are not based on hive format, the order of the `partition_columns` specified columns must match the order as they appear in the object name in the file path specified in the `file_url_list` parameter.

  To see all the `format` parameter options describing the format of the source files, see `DBMS_CLOUD Package Format Options`.

### Usage Notes

- You cannot call this procedure with both `partitioning_clause` and `file_url_list` parameters.

- Specifying the `column_list` parameter is optional with structured data files, including Avro, Parquet, or ORC data files. If `column_list` is not specified, the `format` parameter `partition_columns` option must include both `name` and `type`. 
The `column_list` parameter is required with unstructured data files, such as CSV text files.

The procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` supports external partitioned files in the supported cloud object storage services, including:

- Oracle Cloud Infrastructure Object Storage
- Azure Blob Storage
- Amazon S3
- Amazon S3-Compatible, including: Oracle Cloud Infrastructure Object Storage, Google Cloud Storage, and Wasabi Hot Cloud Storage.
- GitHub Repository

See DBMS_CLOUD Package File URI Formats for more information.

When you call `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` with the `file_url_list` parameter, the types for columns specified in the Cloud Object Store file name must be one of the following types:

- `VARCHAR2(n)`
- `NUMBER(n)`
- `NUMBER(p,s)`
- `NUMBER`
- `DATE`
- `TIMESTAMP(9)`

The default record delimiter is `detected newline`. With `detected newline`, DBMS_CLOUD tries to automatically find the correct newline character to use as the record delimiter. DBMS_CLOUD first searches for the Windows newline character `\r\n`. If it finds the Windows newline character, this is used as the record delimiter for all files in the procedure. If a Windows newline character is not found, DBMS_CLOUD searches for the UNIX/Linux newline character `\n`, and if it finds one it uses `\n` as the record delimiter for all files in the procedure. If the source files use a combination of different record delimiters, you may encounter an error such as, "KUP-04020: found record longer than buffer size supported". In this case, you need to either modify the source files to use the same record delimiter or only specify the source files that use the same record delimiter.

See DBMS_CLOUD Package Format Options for information on the `record_delimiter` format option.

The external partitioned tables you create with `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from.

- `file$path`: Specifies the file path text up to the beginning of the object name.
- `file$name`: Specifies the object name, including all the text that follows the bucket name.

Examples

Example using the `partitioning_clause` parameter:

```sql
BEGIN
  DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
      table_name => 'PET1',
```
Example using the file_uri_list and column_list parameters with unstructured data files:

BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
    table_name => 'MYSALES',
    credential_name => 'DEF_CRED_NAME',
    file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/*.csv',
    column_list => 'product varchar2(100), units number, country varchar2(100), year number, month varchar2(2)',
    field_list => 'product, units', --[Because country, year and month are not in the file, they are not listed in the field list]
    format => '{"type":"csv", "partition_columns": ["country","year","month"]}');
END;
/

Example using the file_uri_list without the column_list parameter with structured data files:

BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
    table_name => 'MYSALES',
    credential_name => 'DEF_CRED_NAME',
    file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/*.parquet',
    format =>
        json_object('type' value 'parquet', 'schema' value 'first',
            'partition_columns' value 'product'),
    location =>
        json_object('type' value 'parquet', 'schema' value 'first',
            'partition_columns' value 'product'),
);
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

```sql
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. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL Procedure for more information.</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>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</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 Avro, ORC, or Parquet format files, see CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet 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
  - Azure Blob Storage
  - Amazon S3
  - Amazon S3-Compatible, including: Oracle Cloud Infrastructure Object Storage, Google Cloud Storage, and Wasabi Hot Cloud Storage.
  - GitHub Repository
  
  The credential is a table level property; therefore, the external files must be on the same object store.
  
  See DBMS_CLOUD Package File URI Formats for more information.

- The default record delimiter is detected newline. With detected newline, DBMS_CLOUD tries to automatically find the correct newline character to use as the record delimiter. DBMS_CLOUD first searches for the Windows newline character \r\n. If it finds the Windows newline character, this is used as the record delimiter for all files in the procedure. If a Windows newline character is not found, DBMS_CLOUD searches for the UNIX/Linux newline character \n, and if it finds one it uses \n as the record delimiter for all files in the procedure. If the source files use a combination of different record delimiters, you may encounter an error such as, "KUP-04020: found record longer than buffer size supported". In this case, you need to either modify the source files to use the same record delimiter or only specify the source files that use the same record delimiter.
  
  See DBMS_CLOUD Package Format Options for information on the recorddelimiter format option.

**Example**

BEGIN  
DBMS_CLOUD.CREATE_EXTERNAL_TABLE(  
  table_name =>'WEATHER_REPORT_DOUBLE_DATE',  
  credential_name =>'OBJ_STORE_CRED',  
  file_uri_list =>'&base_URL/Charlotte_NC_Weather_History_Double_Dates.csv',  
  format => json_object('type' value 'csv', 'skipheaders' value '1'),  
  field_list => 'REPORT_DATE DATE''mm/dd/yy'',  
              REPORT_DATE_COPY DATE ''yyyy-mm-dd'',  
              ACTUAL_MEAN_TEMP'
CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files

This procedure with the format parameter type set to the value avro, orc, or parquet creates an external table with either Avro, ORC, or Parquet format files in the Cloud. This allows you to run queries on external data from Autonomous Database.

Syntax

```sql
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. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL Procedure for more information.</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-&gt;schema parameter which specifies that the schema, columns, and data types, are derived automatically. See the format parameter for details. When the column_list is specified for Avro, ORC, or Parquet source, the column names must match those columns found in the file. Oracle data types must map appropriately to the Avro, ORC, or Parquet data types. For Parquet files, see <a href="#">DBMS_CLOUD Package Parquet to Oracle Data Type Mapping</a> for details. For ORC files, see <a href="#">DBMS_CLOUD Package ORC to Oracle Data Type Mapping</a> for details. For Avro files, see <a href="#">DBMS_CLOUD Package Avro to Oracle Data Type Mapping</a> for details.</td>
</tr>
<tr>
<td>field_list</td>
<td>Ignored for Avro, ORC, or Parquet files. The fields in the source match the external table columns by name. Source data types are converted to the external table column data type. For ORC files, see <a href="#">DBMS_CLOUD Package ORC to Oracle Data Type Mapping</a> for details. For Parquet files, see <a href="#">DBMS_CLOUD Package Parquet to Oracle Data Type Mapping</a> for details. For Avro files, see <a href="#">DBMS_CLOUD Package Avro to Oracle Data Type Mapping</a> for details.</td>
</tr>
<tr>
<td>format</td>
<td>For Avro, ORC, or Parquet, there are only two supported parameters. See <a href="#">DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet</a> for details.</td>
</tr>
</tbody>
</table>

### Examples ORC

```
format => '{"type":"orc", "schema": "all"}"
```

```
format => json_object('type' value 'orc', 'schema' value 'first')
```

### Examples Avro

```
format => '{"type":"avro", "schema": "all"}"
```

```
format => json_object('type' value 'avro', 'schema' value 'first')
```

### Examples Parquet

```
format => '{"type":"parquet", "schema": "all"}"
```

```
format => json_object('type' value 'parquet', 'schema' value 'first')
```
Avro, ORC, or Parquet Column Name Mapping to Oracle Column Names

See DBMS_CLOUD Package Avro, ORC, and Parquet to Oracle Column Name Mapping for information on column name mapping and column name conversion usage in Oracle SQL.

CREATE_HYBRID_PART_TABLE Procedure

This procedure creates a hybrid partitioned table. This allows you to run queries on hybrid partitioned data from Autonomous Database.

Syntax

```sql
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></td>
<td>You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource</td>
</tr>
<tr>
<td></td>
<td>principal is enabled. See ENABLE_RESOURCE_PRINCIPAL Procedure for more</td>
</tr>
<tr>
<td></td>
<td>information.</td>
</tr>
<tr>
<td>partitioning_clause</td>
<td>Specifies the complete partitioning clause, including the location</td>
</tr>
<tr>
<td></td>
<td>information for individual partitions.</td>
</tr>
<tr>
<td>column_list</td>
<td>Comma-delimited list of column names and data types for the external</td>
</tr>
<tr>
<td></td>
<td>table.</td>
</tr>
<tr>
<td>field_list</td>
<td>Identifies the fields in the source files and their data types. The default</td>
</tr>
<tr>
<td></td>
<td>value is NULL meaning the fields and their data types are determined by</td>
</tr>
<tr>
<td></td>
<td>the column_list parameter. This argument's syntax is the same as the</td>
</tr>
<tr>
<td></td>
<td>field_list clause in regular Oracle external tables. For more information</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>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
  - Azure Blob Storage
  - Amazon S3
  - Amazon S3-Compatible, including: Oracle Cloud Infrastructure Object Storage, Google Cloud Storage, and Wasabi Hot Cloud Storage.
– GitHub Repository

The credential is a table level property; therefore, the external files must be on the same object store.

See DBMS_CLOUD Package File URI Formats for more information.

• The external partitioned tables you create with DBMS_CLOUD.CREATE_HYBRID_PART_TABLE include two invisible columns file$path and file$name. These columns help identify which file a record is coming from.
  – file$path: Specifies the file path text up to the beginning of the object name.
  – file$name: Specifies the object name, including all the text that follows the bucket name.

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**

```
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 `adb_user`:

```
GRANT WRITE ON DIRECTORY data_pump_dir TO adb_user;
```

**Example**

```
BEGIN
  DBMS_CLOUD.DELETE_FILE(
    directory_name => 'DATA_PUMP_DIR',
    file_name => 'exp1.dmp');
END;
/
```
DELETE_OBJECT Procedure

This procedure deletes the specified object on object store.

Syntax

```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. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL Procedure for more information.</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>

Example

```sql
BEGIN
    DBMS_CLOUD.DELETE_OBJECT(
        credential_name => 'DEF_CRED_NAME',
        object_uri => 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp1.dmp');
END;
/
```

EXPORT_DATA Procedure

This procedure exports data from Autonomous Database to files in the Cloud based on the result of the specified SQL query. The overloaded form enables you to use the operation_id parameter. Depending on the format type parameter value, the procedure exports rows to the Cloud Object Store as text files in CSV, JSON, or XML format or using the ORACLE_DATAPUMP access driver to write data to a dump file.

Syntax

```sql
DBMS_CLOUD.EXPORT_DATA (    file_uri_list IN CLOB,
    format IN CLOB,
    credential_name IN VARCHAR2 DEFAULT NULL,
    query IN CLOB);  
```

```sql
DBMS_CLOUD.EXPORT_DATA (    file_uri_list IN CLOB DEFAULT NULL,
    format IN CLOB 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. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL Procedure for more information.</td>
</tr>
<tr>
<td>file_uri_list</td>
<td>There are two different forms, depending on the value of the format parameter:</td>
</tr>
<tr>
<td></td>
<td>• When the format parameter type value is json: The JSON on Object Store is saved with a generated file name based on the value of the file_uri_list parameter. See File Naming with Text Output (CSV, JSON, or XML) for more information.</td>
</tr>
<tr>
<td></td>
<td>• When the format parameter type value is datapump, the file_uri_list is a comma-delimited list of the dump files. This specifies the files to be created on the Object Store. Use of wildcard and substitution characters is not supported in the file_uri_list. 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>format</td>
<td>A JSON string that provides export format options. Supported options are:</td>
</tr>
<tr>
<td></td>
<td>• type: The type format option is required and must have one of the values: csv</td>
</tr>
<tr>
<td></td>
<td>• COMPRESSION: The valid values are: BASIC, LOW, MEDIUM, and HIGH.</td>
</tr>
<tr>
<td></td>
<td>• VERSION: The valid values are: COMPATIBLE, LATEST, and a specified version_number.</td>
</tr>
<tr>
<td>query</td>
<td>Use this parameter to specify a SELECT statement so that only the required data is exported. The query determines the contents of the files you export as text files (CSV, JSON, or XML) or dump files. For example: SELECT warehouse_id, quantity FROM inventories For information with the format type value datapump, see Oracle Data Pump Export Data Filters and Unloading and Loading Data with the ORACLE_DATAPUMP Access Driver for more information. When the format type value is json, each query result is checked and if it is not JSON, as determined with the function: JSON_OBJECT_T.parse(),DBMS_CLOUD.EXPORT_DATA transforms the query to include JSON_OBJECT function to convert the row into JSON. See JSON_OBJECT_T Object Type for more information. For example:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```sql
SELECT JSON_OBJECT(* RETURNING CLOB) from(SELECT warehouse_id, quantity FROM inventories)
```
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operation_id</td>
<td>Use this parameter to track the progress and final status of the export operation as the corresponding ID in the USER_LOAD_OPERATIONS view.</td>
</tr>
</tbody>
</table>

#### Usage Notes:

- The `query` parameter value that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.

- Depending on the format parameter specified, `DBMS_CLOUD.EXPORT_DATA` writes the results of the specified query on the Cloud Object Store in one of these formats:
  - JSON files.
    - See [Move Data to Object Store as CSV, JSON, or XML Using EXPORT_DATA](#) for more information on using `DBMS_CLOUD.EXPORT_DATA` with JSON output files.
  - Using the ORACLE_DATAPUMP access driver to write data to a dump file.

#### Usage Notes for ORACLE_DATAPUMP Output (`DBMS_CLOUD.EXPORT_DATA` with format parameter type option `datapump`):

- Autonomous Database export using `DBMS_CLOUD.EXPORT_DATA` with format parameter type option `datapump` only supports Oracle Cloud Infrastructure Object Storage and Oracle Cloud Infrastructure Object Storage Classic object stores.

- Oracle Data Pump divides each dump file part into smaller chunks for faster uploads. The Oracle Cloud Infrastructure Object Storage console shows multiple files for each dump file part that you export. The size of the actual dump files will be displayed as zero (0) and its related file chunks as 10mb or less. For example:

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

  Downloading the zero byte dump file from the Oracle Cloud Infrastructure console or using the Oracle Cloud Infrastructure CLI will not give you the full dump files. To download the full dump files from the Object Store, use a tool that supports Swift such as curl, and provide your user login and Swift auth token.

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

- If you import a file with the `DBMS_CLOUD` procedures that support the format parameter type with the value 'datapump', you only need to provide the primary file name. The procedures that support the 'datapump' format type automatically discover and download the chunks.

- When you use `DBMS_CLOUD.DELETE_OBJECT`, the procedure automatically discovers and deletes the chunks when the procedure deletes the primary file.
• The DBMS_CLOUD.EXPORT_DATA procedure creates the dump file(s) from the file_uri_list values that you specify, as follows:
  – As more files are needed, the procedure creates additional files from the file_uri_list.
  – The procedure does not overwrite files. If a dump file in the file_uri_list exists, DBMS_CLOUD.EXPORT_DATA reports an error.
  – DBMS_CLOUD.EXPORT_DATA does not create buckets.
• The number of dump files that DBMS_CLOUD.EXPORT_DATA generates is determined when the procedure runs. The number of dump files that are generated depends on the number of file names you provide in the file_uri_list parameter, as well as on the number of Autonomous Database OCPUs available to the instance, the service level, and the size of the data.
  For example, if you use a 1 OCPU Autonomous Database instance or the low service, then a single dump file is exported with no parallelism, even if you provide multiple file names. If you use a 4 OCPU Autonomous Database instance with the medium or high service, then the jobs can run in parallel and multiple dump files are exported if you provide multiple file names.
• The dump files you create with DBMS_CLOUD.EXPORT_DATA cannot be imported using Oracle Data Pump impdp. Depending on the database, you can use these files as follows:
  – On an Autonomous Database 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.

Examples

The following example shows DBMS_CLOUD.EXPORT_DATA with the format type parameter with the value datapump:

BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name =>'OBJ_STORE_CRED',
    file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp1.dmp,
    format => json_object('type' value 'datapump', 'compression' value 'basic', 'version' value 'latest'),
    query => 'SELECT warehouse_id, quantity FROM inventories'
  );
END;
/
In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

The following example shows DBMS_CLOUD.EXPORT_DATA with the format type parameter with the value json:

BEGIN
    DBMS_CLOUD.EXPORT_DATA(
        credential_name => 'OBJ_STORE_CRED',
        file_uri_list   => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/json_exportfile,
        query           => 'SELECT * FROM DEPT',
        format          => JSON_OBJECT('type' value 'json', 'compression' value 'gzip'));
    END;
/

The following example shows DBMS_CLOUD.EXPORT_DATA with the format type parameter with the value xml:

BEGIN
    DBMS_CLOUD.EXPORT_DATA(
        credential_name => 'OBJ_STORE_CRED',
        file_uri_list   => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/json_exportfile,
        query           => 'SELECT * FROM DEPT',
        format          => JSON_OBJECT('type' value 'xml', 'compression' value 'gzip'));
    END;
/

The following example shows DBMS_CLOUD.EXPORT_DATA with the format type parameter with the value csv:

BEGIN
    DBMS_CLOUD.EXPORT_DATA(
        credential_name => 'OBJ_STORE_CRED',
        file_uri_list   => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/json_exportfile,
        query           => 'SELECT * FROM DEPT',
        format          => JSON_OBJECT('type' value 'csv', 'delimiter' value '|', 'compression' value 'gzip', 'header' value true ));
    END;
/
GET_OBJECT Procedure and Function

This procedure is overloaded. The procedure form reads an object from Cloud Object Storage and copies it to Autonomous Database. The function form reads an object from Cloud Object Storage and returns a BLOB to Autonomous Database.

Syntax

```sql
DBMS_CLOUD.GET_OBJECT
(credential_name      IN VARCHAR2,
 object_uri           IN VARCHAR2,
 directory_name       IN VARCHAR2,
 file_name            IN VARCHAR2 DEFAULT  NULL,
 startoffset          IN NUMBER DEFAULT  0,
 endoffset            IN NUMBER DEFAULT  0,
 compression          IN VARCHAR2 DEFAULT  NULL);

DBMS_CLOUD.GET_OBJECT
(credential_name      IN VARCHAR2 DEFAULT NULL,
 object_uri           IN VARCHAR2,
 startoffset          IN NUMBER DEFAULT  0,
 endoffset            IN NUMBER DEFAULT  0,
 compression          IN VARCHAR2 DEFAULT  NULL)
RETURN BLOB;
```

Parameters

<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. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL Procedure for more information.</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>
<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 `adb_user`:

```
GRANT WRITE ON DIRECTORY data_pump_dir TO adb_user;
```

Return Values

The function form reads from Object Store and `DBMS_CLOUD.GET_OBJECT` returns a `BLOB`.

Examples

```
BEGIN
  DBMS_CLOUD.GET_OBJECT(
    credential_name => 'OBJ_STORE_CRED',
    object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/file.txt',
    directory_name => 'DATA_PUMP_DIR');
END;
/
```

To read character data from a file in Object Store:

```
SELECT TO_CLOB(
  DBMS_CLOUD.GET_OBJECT(
    credential_name => 'OBJ_STORE_CRED',
    object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/file.txt'))
FROM DUAL;
```

To add an image stored on Object Store in a `BLOB` in the database:

```
DECLARE
  l_blob BLOB := NULL;
BEGIN
  l_blob := DBMS_CLOUD.GET_OBJECT(
    credential_name => 'OBJ_STORE_CRED',
    object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/MyImage.gif');
END;
/
```

In this example, `namespace-string` is the Oracle Cloud Infrastructure object storage namespace and `bucketname` is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.
LIST_FILES Function

This function lists the files in the specified directory. The results include the file names and additional metadata about the files such as file size in bytes, creation timestamp, and the last modification timestamp.

Syntax

```
DBMS_CLOUD.LIST_FILES (
    directory_name      IN VARCHAR2)
RETURN TABLE;
```

Parameters

<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 `adb_user`:

  ```sql
  GRANT READ ON DIRECTORY data_pump_dir TO adb_user;
  ```

- This is a pipelined table function with return type as `DBMS_CLOUD_TYPES.list_object_ret_t`.

- `DBMS_CLOUD.LIST_FILES` does not obtain the checksum value and returns `NULL` for this field.

Example

This is a pipelined function that returns a row for each file. For example, use the following query to use this function:

```
SELECT * FROM DBMS_CLOUD.LIST_FILES('DATA_PUMP_DIR');
```

<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. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL Procedure for more information.</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 timestamp</td>
</tr>
<tr>
<td>Oracle Cloud Infrastructure Swift</td>
<td>Returns NULL</td>
<td>Returns timestamp</td>
</tr>
<tr>
<td>Oracle Cloud Infrastructure Classic</td>
<td>Returns NULL</td>
<td>Returns timestamp</td>
</tr>
<tr>
<td>Amazon S3</td>
<td>Returns NULL</td>
<td>Returns timestamp</td>
</tr>
<tr>
<td>Amazon S3-Compatible</td>
<td>Returns NULL</td>
<td>Returns timestamp</td>
</tr>
<tr>
<td>Azure</td>
<td>Returns timestamp</td>
<td>Returns timestamp</td>
</tr>
<tr>
<td>GitHub Repository</td>
<td></td>
<td></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>2019-11-23T06:36:54Z</td>
<td></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 is overloaded. In one form the procedure copies a file from Autonomous Database to the Cloud Object Storage. In another form the procedure copies a BLOB from Autonomous Database to the Cloud Object Storage.

Syntax

```sql
DBMS_CLOUD.PUT_OBJECT (credential_name IN VARCHAR2,
  object_uri IN VARCHAR2,
  directory_name IN VARCHAR2,
  file_name IN VARCHAR2);

DBMS_CLOUD.PUT_OBJECT (credential_name IN VARCHAR2,
  object_uri IN VARCHAR2,
  contents IN BLOB,
  file_name IN VARCHAR2);
```

Parameters

<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. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLE_RESOURCE_PRINCIPAL Procedure for more information.</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. 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 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 `adb_user`:

```sql
GRANT READ ON DIRECTORY data_pump_dir TO adb_user;
```

**Example**

To handle BLOB data after in-database processing and then store the data directly into a file in the object store:

```sql
DECLARE
    my_blob_data BLOB;
BEGIN
    /* Some processing producing BLOB data and populating my_blob_data */
    DBMS_CLOUD.PUT_OBJECT(
        credential_name => 'OBJ_STORE_CRED',
        object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/my_new_file',
        contents => my_blob_data);
END;
/
```

**Usage Notes**

Depending on your Cloud Object Storage, the size of the object you transfer is limited as follows:

<table>
<thead>
<tr>
<th>Cloud Object Storage Service</th>
<th>Object Transfer Size Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Cloud Infrastructure Object Storage</td>
<td>50 GB</td>
</tr>
<tr>
<td>Amazon S3</td>
<td>5 GB</td>
</tr>
<tr>
<td>Azure Blob Storage</td>
<td>256 MB</td>
</tr>
<tr>
<td>Amazon S3-Compatible</td>
<td>Set by the object store provider. For more information, refer to the provider's documentation.</td>
</tr>
</tbody>
</table>
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.

SYNC_EXTERNAL_PART_TABLE Procedure

This procedure simplifies updating an external partitioned table from files in the Cloud. Run this procedure whenever new partitions are added or when partitions are removed from the Object Store source for the external partitioned table.

Syntax

```
DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE (table_name        IN VARCHAR2,
schema_name       IN VARCHAR2 DEFAULT,
update_columns    IN BOOLEAN DEFAULT);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>The name of the target table. The target table needs to be created before you run <code>DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE</code>.</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>update_columns</td>
<td>The new files may introduce a change to the schema. Updates supported include: new columns, deleted columns. Updates to existing columns, for example a change in the data type throw errors. Default Value: False</td>
</tr>
</tbody>
</table>

VALIDATE_EXTERNAL_PART_TABLE Procedure

This procedure validates the source files for an external partitioned table, generates log information, and stores the rows that do not match the format options specified for the external table in a `badfile` table on Autonomous Database. The overloaded form enables you to use the `operation_id` parameter.

Syntax

```
DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE (table_name                 IN VARCHAR2,
partition_name             IN VARCHAR2 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_EXTERNAL_TABLE Procedure

This procedure validates the source files for an external table, generates log information, and stores the rows that do not match the format options specified for the external table in a badfile table on Autonomous Database. The overloaded form enables you to use the operation_id parameter.

Syntax

```
DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE (  
  table_name      IN VARCHAR2,  
  schema_name     IN VARCHAR2 DEFAULT,  
  rowcount        IN NUMBER DEFAULT,  
  stop_on_error   IN BOOLEAN DEFAULT);  

DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE(  
  table_name     IN VARCHAR2,  
  operation_id   OUT NOCOPY NUMBER,  
  schema_name    IN VARCHAR2 DEFAULT NULL,  
)  
```
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. If the external table refers to Avro, ORC, or Parquet files then the validate stops at the first rejected row. When the external table specifies the format parameter type set to the value avro, orc, or parquet, the parameter stop_on_error effectively always has the value TRUE. Thus, the table badfile will always be empty for an external table referring to Avro, ORC, or Parquet files.</td>
</tr>
</tbody>
</table>

Usage Notes

- DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE works with both partitioned external tables and hybrid partitioned tables. This potentially reads data from all external partitions until rowcount is reached or stop_on_error applies. You do not have control over which partition, or parts of a partition, is read in which order.

VALIDATE_HYBRID_PART_TABLE Procedure

This procedure validates the source files for a hybrid partitioned table, generates log information, and stores the rows that do not match the format options specified for the hybrid table in a badfile table on Autonomous Database. The overloaded form enables you to use the operation_id parameter.

Syntax

```sql
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);  
```
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>

DBMS_CLOUD REST APIs

This section covers the DBMS_CLOUD REST APIs provided with Autonomous Database.

<table>
<thead>
<tr>
<th>REST API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET_RESPONSE_HEADERS Function</td>
<td>This function returns the HTTP response headers as JSON data in a JSON object in Autonomous Database.</td>
</tr>
<tr>
<td>GET_RESPONSE_RAW Function</td>
<td>This function returns the HTTP response in RAW format Autonomous Database. This is useful if the HTTP response is expected to be binary format.</td>
</tr>
<tr>
<td>GET_RESPONSE_STATUS_CODE Function</td>
<td>This function returns the HTTP response status code as an integer in Autonomous Database. The status code helps to identify if the request is successful.</td>
</tr>
<tr>
<td>GET_RESPONSE_TEXT Function</td>
<td>This function returns the HTTP response in TEXT format (VARCHAR2 or CLOB) in Autonomous Database. Usually, most Cloud REST APIs return JSON response in text format. This function is useful if you expect the the HTTP response is in text format.</td>
</tr>
<tr>
<td>GET_API_RESULT_CACHE_SIZE Function</td>
<td>This function returns the configured result cache size.</td>
</tr>
</tbody>
</table>
### REST API

<table>
<thead>
<tr>
<th>REST API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SEND_REQUEST Function and Procedure</strong></td>
<td>This function begins an HTTP request, gets the response, and ends the response in Autonomous Database. This function provides a workflow for sending a Cloud REST API request with arguments and a return response code and payload.</td>
</tr>
<tr>
<td><strong>SET_API_RESULT_CACHE_SIZE Procedure</strong></td>
<td>This procedure sets the maximum cache size for current session.</td>
</tr>
</tbody>
</table>

### Topics

- DBMS_CLOUD REST API Overview
- DBMS_CLOUD REST API Constants
- DBMS_CLOUD REST API Results Cache
- DBMS_CLOUD REST API Examples

### DBMS_CLOUD REST API Overview

When you use PL/SQL in your application and you need to call Cloud REST APIs you can use `DBMS_CLOUD.SEND_REQUEST` to send the REST API requests.

The `DBMS_CLOUD` REST API functions allow you to make HTTP requests using `DBMS_CLOUD.SEND_REQUEST` and obtain and save results. These functions provide a generic API that lets you call any REST API with the following supported cloud services:

- **Oracle Cloud Infrastructure**
  See [API Reference and Endpoints](#) for information on Oracle Cloud Infrastructure REST APIs.

- **Amazon Web Services (AWS)**
  See [Guides and API References](#) for information on Amazon Web Services REST APIs.

- **Azure Cloud**
  See [Azure REST API Reference](#) for information on Azure REST APIs.

- **Oracle Cloud Infrastructure Classic**
  See [All REST Endpoints](#) for information on Oracle Cloud Infrastructure Classic REST APIs.

- **GitHub Repository**
  See [GitHub REST API](#) for more information.

### DBMS_CLOUD REST API Constants

Describes the `DBMS_CLOUD` constants for making HTTP requests using `DBMS_CLOUD.SEND_REQUEST`.

`DBMS_CLOUD` supports GET, PUT, POST, HEAD and DELETE HTTP methods. The REST API method to be used for an HTTP request is typically documented in the Cloud REST API documentation.

---

1 Support for Azure Cloud REST API calls is limited to the domain "blob.windows.net".
<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>
<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>

DBMS_CLOUD REST API Results Cache

You can save DBMS_CLOUD REST API results when you set the cache parameter to true with DBMS_CLOUD.SEND_REQUEST. The SESSION_CLOUD_API_RESULTS view describes the columns you can use when REST API results are saved.

By default DBMS_CLOUD REST API calls do not save results for your session. In this case you use the DBMS_CLOUD.SEND_REQUEST function to return results.

When you use DBMS_CLOUD.SEND_REQUEST and set the cache parameter to TRUE, results are saved and you can view past results in the SESSION_CLOUD_API_RESULTS view. Saving and querying historical results of DBMS_CLOUD REST API requests can help you when you need to work with your previous results in your applications.

For example, to query recent DBMS_CLOUD REST API results, use the view SESSION_CLOUD_API_RESULTS:

SELECT timestamp FROM SESSION_CLOUD_API_RESULTS;

When you save DBMS_CLOUD REST API results with DBMS_CLOUD.SEND_REQUEST the saved data is only available within the same session (connection). After the session exits, the saved data is no longer available.

Use DBMS_CLOUD.GET_API_RESULT_CACHE_SIZE and DBMS_CLOUD.SET_API_RESULT_CACHE_SIZE to view and set the DBMS_CLOUD REST API cache size, and to disable caching.

DBMS_CLOUD REST API Results cache_scope Parameter

When you save DBMS_CLOUD REST API results with DBMS_CLOUD.SEND_REQUEST, access to the results in SESSION_CLOUD_API_RESULTS is provided based on the value of cache_scope.

By default cache_scope is 'PRIVATE' and only the current user of the session can access the results. If you set the cache_scope to 'PUBLIC', then all session users can access the results. The default value for cache_scope specifies that each user can only see DBMS_CLOUD.SEND_REQUEST REST API results generated by the procedures they invoke with invoker's rights. When you invoke DBMS_CLOUD.SEND_REQUEST in a session, there are three possibilities that determines if the current user can see results in the cache, based on the cache_scope value:

- You directly execute DBMS_CLOUD.SEND_REQUEST as a top-level statement and the call to DBMS_CLOUD.SEND_REQUEST and the REST API results are saved with the same username. In this case you have access to all results with the default value, 'PRIVATE', set for cache_scope.
• You write a wrapper invoker’s rights procedure and as the current user your call with DBMS_CLOUD.SEND_REQUEST calls the procedure and the REST API results are saved with the same username. In this case, and you have access to all results with the default value, 'PRIVATE', set for cache_scope.

• You write a wrapper definer’s rights procedure and the procedure is owned by another user. When you call DBMS_CLOUD.SEND_REQUEST inside the procedure, the results are saved with the username of the procedure owner.

For this case, a different definer’s rights user is invoking DBMS_CLOUD.SEND_REQUEST, and the REST API results are saved with that definer’s procedure’s owner. For this case, by default when cache_scope is 'PRIVATE', the invoker’s session cannot see the results.

If the definer’s procedure owner wants to make the results available to any invoking session user, then they must set cache_scope to 'PUBLIC' in the DBMS_CLOUD.SEND_REQUEST.

**DBMS_CLOUD REST API SESSION_CLOUD_API_RESULTS View**

You can save DBMS_CLOUD REST API results when you set the cache parameter to true with DBMS_CLOUD.SEND_REQUEST. The SESSION_CLOUD_API_RESULTS view describes the columns you can use when REST API results are saved.

The view SESSION_CLOUD_API_RESULTS is the view created if you cache results with DBMS_CLOUD.SEND_REQUEST. You can query historical results which belong to your user session. When the session ends, the data in the SESSION_CLOUD_API_RESULTS is purged.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URI</td>
<td>The DBMS_CLOUD REST API request URL</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>The DBMS_CLOUD REST API response timestamp</td>
</tr>
<tr>
<td>CLOUD_TYPE</td>
<td>The DBMS_CLOUD REST API cloud type, such as Oracle Cloud Infrastructure, AMAZON_S3, and AZURE_BLOB</td>
</tr>
<tr>
<td>REQUEST_METHOD</td>
<td>The DBMS_CLOUD REST API request method, such as GET, PUT, HEAD</td>
</tr>
<tr>
<td>REQUEST_HEADERS</td>
<td>The DBMS_CLOUD REST API request headers</td>
</tr>
<tr>
<td>REQUEST_BODY_TEXT</td>
<td>The DBMS_CLOUD REST API request body in CLOB</td>
</tr>
<tr>
<td>RESPONSE_STATUS_CODE</td>
<td>The DBMS_CLOUD REST API response status code, such as 200 (OK), 404 (Not Found)</td>
</tr>
<tr>
<td>RESPONSE_HEADERS</td>
<td>The DBMS_CLOUD REST API response headers</td>
</tr>
<tr>
<td>RESPONSE_BODY_TEXT</td>
<td>The DBMS_CLOUD REST API response body in CLOB</td>
</tr>
<tr>
<td>SCOPE</td>
<td>The cache_scope set by DBMS_CLOUD.SEND_REQUEST. Valid values are PUBLIC or PRIVATE.</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_RAW_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

```
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**

```sql
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>

GET_API_RESULT_CACHE_SIZE Function

This function returns the configured result cache size. The cache size value only applies for the current session.

**Syntax**

```sql
DBMS_CLOUD.GET_API_RESULT_CACHE_SIZE()
    RETURN NUMBER;
```
SEND_REQUEST Function and Procedure

This function and procedure begins an HTTP request, gets the response, and ends the response. This function provides a workflow for sending a cloud REST API request with arguments and the function returns a response code and payload. If you use the procedure, you can view results and response details from the saved results with the SESSION_CLOUD_API_RESULTS view.

Syntax

```
DBMS_CLOUD.SEND_REQUEST
(credential_name IN VARCHAR2,
  uri IN VARCHAR2,
  method IN VARCHAR2,
  headers IN CLOB DEFAULT NULL,
  async_request_url IN VARCHAR2 DEFAULT NULL,
  wait_for_states IN DBMS_CLOUD_TYPES.wait_for_states_t DEFAULT NULL,
  timeout IN NUMBER DEFAULT 0,
  cache IN PL/SQL BOOLEAN DEFAULT FALSE,
  cache_scope IN VARCHAR2 DEFAULT 'PRIVATE',
  body IN BLOB DEFAULT NULL)
RETURN DBMS_CLOUD_TYPES.resp;
```

```
DBMS_CLOUD.SEND_REQUEST
(credential_name IN VARCHAR2,
  uri IN VARCHAR2,
  method IN VARCHAR2,
  headers IN CLOB DEFAULT NULL,
  async_request_url IN VARCHAR2 DEFAULT NULL,
  wait_for_states IN DBMS_CLOUD_TYPES.wait_for_states_t DEFAULT NULL,
  timeout IN NUMBER DEFAULT 0,
  cache IN PL/SQL BOOLEAN DEFAULT FALSE,
  cache_scope IN VARCHAR2 DEFAULT 'PRIVATE',
  body IN BLOB DEFAULT NULL);
```

Parameters

<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. You can use 'OCI$RESOURCE_PRINCIPAL' as the credential_name when resource principal is enabled. See ENABLERESOURCEPRINCIPAL Procedure for more information.</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>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>headers</td>
<td>HTTP Request headers for the corresponding cloud native API in JSON format. The authentication headers are set automatically, only pass custom headers.</td>
</tr>
<tr>
<td>async_request_url</td>
<td>An asynchronous request URL. To obtain the URL select your request API from the list of APIs (see <a href="https://docs.cloud.oracle.com/en-us/aas/api/">https://docs.cloud.oracle.com/en-us/aas/api/</a>). Then, navigate to find the API for your request in the left pane. For example, Database Services API → Autonomous Database → StopAutonomousDatabase. This page shows the API home (and shows the base endpoint). Then, append the base endpoint with the relative path obtained for your work request WorkRequest link.</td>
</tr>
<tr>
<td>wait_for_states</td>
<td>Wait for states is a status of type: DBMS_CLOUD_TYPES.wait_for_states_t. The following are valid values for expected states: 'ACTIVE', 'CANCELED', 'COMPLETED', 'DELETED', 'FAILED', 'SUCCEEDED'. Multiple states are allowed for wait_for_states. The default value for wait_for_states is to wait for any of the expected states: 'ACTIVE', 'CANCELED', 'COMPLETED', 'DELETED', 'FAILED', 'SUCCEEDED'.</td>
</tr>
<tr>
<td>timeout</td>
<td>Specifies the timeout, in seconds, for asynchronous requests with the parameters async_request_url and wait_for_states. Default value is 0. This indicates to wait for completion of the request without any timeout.</td>
</tr>
<tr>
<td>cache</td>
<td>If TRUE specifies the request should be cached in REST result API cache. The default value is FALSE, which means REST API requests are not cached.</td>
</tr>
<tr>
<td>cache_scope</td>
<td>Specifies whether everyone can have access to this request result cache. Valid values: &quot;PRIVATE&quot; and &quot;PUBLIC&quot;. The default value is &quot;PRIVATE&quot;.</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 DBMS_CLOUD.SEND_REQUEST is invalid.</td>
</tr>
<tr>
<td>invalid_req_header</td>
<td>ORA-20024</td>
<td>Request headers passed to DBMS_CLOUD.SEND_REQUEST are not in valid JSON format.</td>
</tr>
</tbody>
</table>

### Usage Notes

- If you are using Oracle Cloud Infrastructure, you must use a Signing Key based credential value for the credential_name. See CREATE_CREDENTIAL Procedure for more information.
- The optional parameters async_request_url, wait_for_states, and timeout allow you to handle long running requests. Using this asynchronous form of send_request, the
function waits for the completion status specified in wait_for_states before returning. With these parameters in the send request, you pass the expected return states in the wait_for_states parameter, and you use the async_request_url parameter to specify an associated work request, the request does not return immediately. Instead, the request probes the async_request_url until the return state is one of the expected states or the timeout is exceeded (timeout is optional). If no timeout is specified, the request waits until a state found in wait_for_states occurs.

SET_API_RESULT_CACHE_SIZE Procedure

This procedure sets the maximum cache size for current session. The cache size value only applies for the current session.

Syntax

```
DBMS_CLOUD.SET_API_RESULT_CACHE_SIZE(
    cache_size          IN NUMBER);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache_size</td>
<td>Set the maximum cache size to the specified value cache_size. If the new maximum cache size is smaller than the current cache size, older records are dropped until the number of rows is equal to the specified maximum cache size. The maximum value is 10000. If the cache size is set to 0, caching is disabled in the session. The default cache size is 10.</td>
</tr>
</tbody>
</table>

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid API result cache size</td>
<td>ORA-20032</td>
<td>The minimum value is 0 and the maximum value is 10000. This exception is shown when the input value is less than 0 or is larger than 10000.</td>
</tr>
</tbody>
</table>

Example

```
EXEC DBMS_CLOUD.SET_API_RESULT_CACHE_SIZE(101);
```

DBMS_CLOUD REST API Examples

Shows examples using DBMS_CLOUD.SEND_REQUEST to create and delete an Oracle Cloud Infrastructure Object Storage bucket, and an example to list all compartments in the tenancy.
Note:

These examples show Oracle Cloud Infrastructure request APIs and require that you use a Signing Key based credential for the `credential_name`. Oracle Cloud Infrastructure Signing Key based credentials include the `private_key` and `fingerprint` arguments.

For example:

```sql
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL (
        credential_name => 'OCI_KEY_CRED',
        user_ocid       => 'ocid1.user.oc1..aaaaaaaauq54mi7zdyfhw33ozkwuontjceel7fok5nq3bf2vwetk
                      pqsoa',
        tenancy_ocid    => 'ocid1.tenancy.oc1..aabbbbbbaafcue47pqmrf4vigneebgmnoy5r7xvoypicjqq
                      ge32ewnrcyx2a',
        private_key     => 'MIIEogIBAAKCAQEAtUnxbmrekwgVac6PdWrzoXvIpaA9+0r1.....wt
                      nNpESQQQQLGP
                      DBNM//JEbg=',
END;
/
```

See CREATE_CREDENTIAL Procedure for information on DBMS_CLOUD.CREATE_CREDENTIAL.

Create Bucket Example

Shows an example using `DBMS_CLOUD.SEND_REQUEST` with HTTP POST method to create an object store bucket named `bucketname`.

See CreateBucket for details on the Oracle Cloud Infrastructure Object Storage Service API for this example.

```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/',
        method => DBMS_CLOUD.METHOD_POST,
        body => UTL_RAW.cast_to_raw(
            JSON_OBJECT('name' value 'bucketname',
                        'compartmentId' value 'compartment_OCID'))
    );
END;
/
-- Response Body in TEXT format
dbms_output.put_line('Body: ' || '----------' || CHR(10) ||
DBMS_CLOUD.get_response_text(resp) || CHR(10)));

-- Response Headers in JSON format
dbms_output.put_line('Headers: ' || CHR(10) || '----------' ||
CHR(10) ||
DBMS_CLOUD.get_response_headers(resp).to_clob || CHR(10)));

-- Response Status Code
dbms_output.put_line('Status Code: ' || CHR(10) || '----------' ||
CHR(10) ||
DBMS_CLOUD.get_response_status_code(resp));

END;
/

Notes:

• In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

• Where: region is an endpoint region. See Object Storage API reference in API Reference and Endpoints for more information. For example, where region is: us-phoenix-1.

Delete Bucket Example

Shows an example using DBMS_CLOUD.SEND_REQUEST with HTTP DELETE method to delete an object store bucket named bucketname.

See DeleteBucket for details on the Oracle Cloud Infrastructure Object Storage Service API for this example.

SET SERVEROUTPUT ON
DECLARE
    resp DBMS_CLOUD_TYPES.resp;
BEGIN
    -- Send request
    resp := DBMS_CLOUD.send_request(
        credential_name => 'OCI_KEY_CRED',
        uri => 'https://objectstorage.
region
.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)));

Appendix A
DBMS_CLOUD Package
-- Response Status Code
   dbms_output.put_line('Status Code: ' || CHR(10) || '------------' ||
   CHR(10) ||
   DBMS_CLOUD.get_response_status_code(resp));

END;
/

Notes:

• In this example, namespace-string is the Oracle Cloud Infrastructure object storage
  namespace and bucketname is the bucket name. See Understanding Object Storage
  Namespaces for more information.

• Where: region is an endpoint region. See Object Storage API reference in API
  Reference and Endpoints for more information. For example, where region is: us-
  phoenix-1.

List Compartments Example

Shows an example using DBMS_CLOUD.SEND_REQUEST with HTTP GET method to list all
compartments in the tenancy (root compartment). This example shows how to pass request
headers in the DBMS_CLOUD.SEND_REQUEST.

See ListCompartments for details on the Oracle Cloud Infrastructure Identity and Access
Management Service API for this example.

--
-- List compartments
--

DECLARE
   resp DBMS_CLOUD_TYPES.resp;
   root_compartment_ocid VARCHAR2(512) := '&1';
BEGIN
   -- Send request
   dbms_output.put_line('Send Request');
   resp := DBMS_CLOUD.send_request(
      credential_name => 'OCI_KEY_CRED',
      uri => 'https://identity.
      region
      .oraclecloud.com/20160918/
      compartments?compartmentId=' || root_compartment_ocid,
      method => DBMS_CLOUD.METHOD_GET,
      headers => JSON_OBJECT('opc-request-id' value 'list-
      compartments')
   );
   dbms_output.put_line('Body: ' || '------------' || CHR(10) ||
   DBMS_CLOUD.get_response_text(resp) || CHR(10));
   dbms_output.put_line('Headers: ' || CHR(10) || '------------' || CHR(10) ||
   DBMS_CLOUD.get_response_headers(resp).to_clob || CHR(10));
   dbms_output.put_line('Status Code: ' || CHR(10) || '------------' ||
   DBMS_CLOUD.get_response_status_code(resp));
END;
/

Appendix A

DBMS_CLOUD Package

A-51
Where: region is an endpoint region. See Identity and Access Management (IAM) API reference in API Reference and Endpoints for more information. For example, where region is: uk-london-1.

Asynchronous Request Example

Shows an example using DBMS_CLOUD SEND_REQUEST with HTTP POST method to perform the Autonomous Database stop operation and wait for status. This example shows how to use DBMS_CLOUD SEND_REQUEST with the async_request_url, wait_for_states, and timeout parameters.

```
-- Sent Work Request Autonomous Database Stop Request with Wait for Status
DECLARE
  l_resp DBMS_CLOUD_TYPES.resp;
  l_resp_json JSON_OBJECT_T;
  l_key_shape JSON_OBJECT_T;
  l_body JSON_OBJECT_T;
  status_array DBMS_CLOUD_TYPES.wait_for_states_t;
BEGIN
  status_array := DBMS_CLOUD_TYPES.wait_for_states_t('SUCCEEDED');
  l_body := JSON_OBJECT_T('{}');
  l_body.put('autonomousDatabaseId', 'ocid');
  -- Send request
  dbms_output.put_line(l_body.to_clob);
  dbms_output.put_line('Send Request');
  l_resp := DBMS_CLOUD.send_request(
    credential_name    => 'NATIVE_CRED_OCI',
    uri                => 'https://database.region.oraclecloud.com/20160918/autonomousDatabases/ocid/actions/stop',
    method             => DBMS_CLOUD.METHOD_POST,
    body               => UTL_RAW.cast_to_raw(l_body.to_clob),
    async_request_url  => 'https://iaas.region.oraclecloud.com/20160918/workRequests',
    wait_for_states    => status_array,
    timeout            => 600
  );
  dbms_output.put_line('resp body: ' ||
    DBMS_CLOUD.get_response_text(l_resp));
  dbms_output.put_line('resp headers: ' ||
    DBMS_CLOUD.get_response_headers(l_resp).to_clob);
END;
/
```

Where: region is an endpoint region. See Identity and Access Management (IAM) API reference in API Reference and Endpoints for more information. For example, where region is: uk-london-1.

The ocid is the Oracle Cloud Infrastructure resource identifier. See Resource Identifiers for more information.
DBMS_CLOUD 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.

Topics

- Oracle Cloud Infrastructure Object Storage Native URI Format
- Oracle Cloud Infrastructure Object Storage Swift URI Format
- Oracle Cloud Infrastructure Object Storage URI Format Using Pre-Authenticated Request URL
- URI Format Using Public URL
- Oracle Cloud Infrastructure Object Storage Classic URI Format
- Amazon S3 URI Format
- Azure Blob Storage URI Format
- Amazon S3 Compatible URI Format
- GitHub Raw URL Format

Oracle Cloud Infrastructure Object Storage Native URI Format

If your source files reside on the Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure native URIs, with the format:

https://objectstorage.region.oraclecloud.com/n/namespace-string/b/bucket/o/filename

For example, the Native URI for the file channels.txt in the bucketname bucket in the Phoenix data center is:

https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/channels.txt

In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

You can find the URI from the Oracle Cloud Infrastructure Object Storage “Object Details” in the right hand side ellipsis menu in the Object Store:

1. Open the Oracle Cloud Infrastructure Console by clicking the □ next to Oracle Cloud.
2. From the Oracle Cloud Infrastructure left navigation menu click Core Infrastructure. Under Object Storage, click Object Storage.
3. Under List Scope, select a Compartment.
4. From the Name column, select a bucket.
5. In the Objects area, click View Object Details.

6. On the Object Details page, the URL Path (URI) field shows the URI to access the object.

![View Object Details](image)

**Note:**

The source files need to be stored in an Object Storage tier bucket. Autonomous Database does not support buckets in the Archive Storage tier. See Overview of Object Storage for more information.

Oracle Cloud Infrastructure Object Storage Swift URI Format

If your source files reside on the Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure Swift URIs with the format:

https://swiftobjectstorage.region.oraclecloud.com/v1/namespace-string/bucket/filename

For example, the Swift URI for the file channels.txt in the bucketname bucket in the Phoenix data center is:

https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/namespace-string/bucketname/channels.txt

In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.
Note:
The source files need to be stored in an Object Storage tier bucket. Autonomous Database does not support buckets in the Archive Storage tier. See Overview of Object Storage for more information.

Oracle Cloud Infrastructure Object Storage URI Format Using Pre-Authenticated Request URL

If your source files reside on the Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure pre-authenticated URIs. When you create a pre-authenticated request, a unique URL is generated. You can then provide the unique URL to users in your organization, partners, or third parties to access the Object Storage resource target identified in the pre-authenticated request.

Note:
Carefully assess the business requirement for and the security ramifications of pre-authenticated access. When you create the pre-authenticated request URL, note the **Expiration** and the **Access Type** to make sure they are appropriate for your use. A pre-authenticated request URL gives anyone who has the URL access to the targets identified in the request for as long as the request is active. In addition to considering the operational needs of pre-authenticated access, it is equally important to manage its distribution.

The format for pre-authenticated request URLs is:

https://objectstorage.region.oraclecloud.com/p/encrypted_string/n/namespace-string/b/bucket/o/filename

For example, a sample pre-authenticated URI for the file `channels.txt` in the `bucketname` bucket in the Phoenix data center is:

https://objectstorage.us-phoenix-1.oraclecloud.com/p/2xN-uDtWJNaiD910UCYGue/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;
/

**Note:**

A list of mixed URLs is valid. If the URL list contains both pre-authenticated URLs and URLs that require authentication, DBMS_CLOUD uses the specified credential_name to access the URLs that require authentication and for the pre-authenticated URLs the specified credential_name is ignored.

See Using Pre-Authenticated Requests for more information.

### URI Format Using Public URL

If your source files reside on an Object Store that provides public URLs, you can use public URLs with DBMS_CLOUD procedures. Public means the Object Storage service supports anonymous, unauthenticated access to the Object Store files. See your Cloud Object Storage service for details on how to make an object public in a supported Object Store.

**Note:**

Carefully assess the business requirement for and the security ramifications of using public URLs. When you use public URLs, due to the file content not being authenticated, make sure this is appropriate for your use.

You can use a public URL in any DBMS_CLOUD procedure that takes a URL to access files in your object store, without the need to create a credential. You need to either specify the credential_name parameter as NULL or not supply a credential_name parameter.

For example the following uses DBMS_CLOUD.COPY_DATA without a credential_name:

BEGIN

DBMS_CLOUD.COPY_DATA(
    table_name => 'CHANNELS',
    file_uri_list => 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/bucketname/o/
    chan_v3.dat',
    format => json_object('delimiter' value ',') );
END;
/

---

**ORACLE**

A-56
In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

Note:

A list of mixed URLs is valid. If the URL list contains both public URLs and URLs that require authentication, DBMS_CLOUD uses the specified credential_name to access the URLs that require authentication and for the public URLs the specified credential_name is ignored.

See Public Buckets for information on using Oracle Cloud Infrastructure public buckets.

Oracle Cloud Infrastructure Object Storage Classic URI Format

If your source files reside in Oracle Cloud Infrastructure Object Storage Classic, see the REST page for a description of the URI format for accessing your files: About REST URLs for Oracle Cloud Infrastructure Object Storage Classic Resources.

Amazon S3 URI Format

If your source files reside in Amazon S3, see the following for a description of the URI format for accessing your files: Accessing a bucket.

For example the following refers to the file channels.txt in the adb bucket in the us-west-2 region.

https://s3-us-west-2.amazonaws.com/adb/channels.txt

You can use a presigned URL in any DBMS_CLOUD procedure that takes a URL to access files in Amazon S3 object store, without the need to create a credential. To use a presigned URL in any DBMS_CLOUD procedure, either specify the credential_name parameter as NULL, or do not supply a credential_name parameter.

See Share an Object with Others for more information.

Note:

DBMS_CLOUD supports the standard Amazon S3 endpoint syntax to access your buckets. DBMS_CLOUD does not support Amazon S3 legacy endpoints. See Legacy Endpoints for more information.

Azure Blob Storage URI Format

If your source files reside in Azure Blob Storage, see the following for a description of the URI format for accessing your files: Resource URI Syntax.
For example the following refers to the file `channels.txt` in the `adb` container in the storage account `adb_user`:

https://adb_user.blob.core.windows.net/adb/channels.txt

---

**Note:**

You can use Shared Access Signatures (SAS) URL in any `DBMS_CLOUD` procedure that takes a URL to access files in Azure Blob Storage, without the need to create a credential. To use a Shared Access Signature (SAS) URL, either specify the `credential_name` parameter as `NULL`, or do not supply a `credential_name` parameter. See Grant Limited Access to Azure Storage Resources Using Shared Access Signatures (SAS) for more information.

---

**Amazon S3 Compatible URI Format**

`DBMS_CLOUD` supports object storage service implementations that support Amazon S3 compatible URLs, including the following services:

- Oracle Cloud Infrastructure Object Storage with Amazon S3 compatible URL
- Google Cloud Storage with Amazon S3 compatible URL
- Wasabi Hot Cloud Storage with Amazon S3 compatible URL

---

**Note:**

To use `DBMS_CLOUD` with an Amazon S3 compatible object store you need to provide valid credentials. See `CREATE_CREDENTIAL` Procedure for more information.

---

If your source files reside on a service that supports Amazon S3 compatible URIs, use the following URI format to access your files:

- **Oracle Cloud Infrastructure Object Storage S3 Compatible URL**
  
  **Object URL Format:**
  
  https://mynamespace.compat.objectstorage.region.oraclecloud.com/bucket_name/object_name
  
  **Bucket URL Format:**
  
  https://mynamespace.compat.objectstorage.region.oraclecloud.com/bucket_name
  
  See Amazon S3 Compatibility and Object Storage Service API for more information.
• **Google Cloud Storage S3 Compatible URL**
  Object URL Format:
  
  https://bucketname.storage.googleapis.com/object_name

  Bucket URL Format:
  
  https://bucketname.storage.googleapis.com/

  See [Migrating from Amazon S3 to Cloud Storage](#) and [Request Endpoints](#) for more information.

• **Wasabi S3 Compatible URL**
  Object URL Format:
  
  https://bucketname.s3.region.wasabisys.com/object_name

  Bucket URL Format:
  
  https://bucketname.s3.region.wasabisys.com/

  See [S3-compatible API Connectivity for Wasabi Hot Cloud Storage](#) and [What are the service URLs for Wasabi's different regions?](#) for more information.

**GitHub Raw URL Format**

DBMS_CLOUD supports GitHub Raw URLs to access data from a GitHub Repository.

**Note:**

For DBMS_CLOUD access with GitHub Raw URLs, repository access is limited to read-only functionality. The DBMS_CLOUD APIs such as `DBMS_CLOUD.PUT_OBJECT` that write data are not supported with DBMS_CLOUD APIs on a GitHub Repository. As an alternative, use `DBMS_CLOUD_REPO.PUT_FILE` to upload data to a GitHub Repository.

Use GitHub Raw URLs with DBMS_CLOUD APIs to access source files that reside on a GitHub Repository. When you browse to a file on GitHub and click the Raw link, this shows the GitHub Raw URL. See the following for a description of GitHub Raw URL format: [What do raw.githubusercontent.com URLs represent?](#).

For example, using `DBMS_CLOUD.GET_OBJECT`:

BEGIN
DBMS_CLOUD.GET_OBJECT(
    credential_name => 'MY_CRED',
    directory_name => 'DATA_PUMP_DIR'
)
For example, using DBMS_CLOUD.CREATE_EXTERNAL_TABLE:

```sql
BEGIN
  DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
    credential_name => 'MY_CRED',
    table_name      => 'EMPLOYEES_EXT',
    column_list     => 'name varchar2(30), gender varchar2(30), salary number',
    format          => JSON_OBJECT('type' value 'csv')
  );
END;
/
```

SELECT * FROM employees_ext;

DBMS_CLOUD procedures that take a URL to access a GitHub Repository do not require credentials with public visibility GitHub repositories. To use a public visibility URL you can specify the credential_name parameter as NULL or not supply a credential_name parameter. See Setting repository visibility for more information.

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:

```sql
format => '{"format_option" : "format_value"}'
```

And:

```sql
format => json_object('format_option' value 'format_value'))
```

Examples:

```sql
format => json_object('type' VALUE 'CSV')
```

To specify multiple format options, separate the values with a ",".

For example:

```sql
format => json_object('ignoremissingcolumns' value 'true', 'removequotes' value 'true',
  'dateformat' value 'YYYY-MM-DD-HH24-MI-SS',
  'blankasnull' value 'true')
```

Note:

For Avro, ORC, or Parquet format options, see DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet.
As noted in the **Format Option** column, a limited set of format options are valid with `DBMS_CLOUD.COPY_COLLECTION` or `DBMS_CLOUD.COPY_DATA` when the format type is `JSON`.

<table>
<thead>
<tr>
<th>Format Option</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>blankasnull</td>
<td>When set to <strong>true</strong>, loads fields consisting of spaces as <strong>null</strong>.</td>
<td>blankasnull: true</td>
</tr>
<tr>
<td></td>
<td><strong>Default value:</strong> <strong>False</strong></td>
<td></td>
</tr>
<tr>
<td>characterset</td>
<td>Specifies the characterset of source files</td>
<td>characterset: <strong>string</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Valid with format JSON and COPY_DATA</strong></td>
<td><strong>Default value:</strong> Database characterset</td>
</tr>
<tr>
<td>columnpath</td>
<td>Array of JSON path expressions that correspond to the fields that need to</td>
<td>JSON Array of json path expressions expressed in string format. For</td>
</tr>
<tr>
<td></td>
<td>be extracted from the JSON records. Each of the JSON path expressions in</td>
<td>example: 'columnpath' value</td>
</tr>
<tr>
<td></td>
<td>the array should follow the rules described in SQL/JSON Path Expressions.</td>
<td>'[$.WEATHER_STATION_ID',</td>
</tr>
<tr>
<td></td>
<td><strong>Only use with format JSON and COPY_DATA</strong></td>
<td>'$.WEATHER_STATION_NAME']'</td>
</tr>
<tr>
<td></td>
<td><strong>Only use with format JSON and DBMS_CLOUD.COPY_DATA.</strong></td>
<td></td>
</tr>
<tr>
<td>compression</td>
<td>Specifies the compression type of the source file. ZIP archiving format is</td>
<td>compression: auto</td>
</tr>
<tr>
<td></td>
<td><strong>Option valid with JSON data</strong></td>
<td>bzip2</td>
</tr>
<tr>
<td></td>
<td><strong>Specifying the value auto checks for the compression types:</strong> gzip, zlib, bzip2.</td>
<td><strong>Default value:</strong> Null value meaning no compression.</td>
</tr>
<tr>
<td>conversionerrors</td>
<td>If a row is rejected because of data type conversion errors, the related</td>
<td>conversionerrors:</td>
</tr>
<tr>
<td></td>
<td>columns are stored as null or the row is rejected.</td>
<td>**reject_record</td>
</tr>
<tr>
<td></td>
<td><strong>Default value:</strong> <strong>reject_record</strong></td>
<td></td>
</tr>
<tr>
<td>dateformat</td>
<td>Specifies the date format in the source file. The format option <strong>AUTO</strong></td>
<td>dateformat: <strong>string</strong></td>
</tr>
<tr>
<td></td>
<td>searches for the following formats:</td>
<td><strong>Default value:</strong> Database date format</td>
</tr>
<tr>
<td></td>
<td>J</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM-DD-YYYYBC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM-DD-YYYY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYYMMDD HHMISS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYYMMDD HHMISS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYY.DDD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYY-MM-DD</td>
<td></td>
</tr>
<tr>
<td>delimiter</td>
<td>Specifies the field delimiter.</td>
<td>delimiter: <strong>character</strong></td>
</tr>
<tr>
<td></td>
<td><strong>To use a special character as the delimiter, specify the HEX value of</strong></td>
<td><strong>Default value:</strong></td>
</tr>
<tr>
<td></td>
<td><strong>the ASCII code of the character.</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>For example, the following specifies the TAB character as the delimiter:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>format =&gt; json_object('delimiter' value 'X''9''')</td>
<td></td>
</tr>
<tr>
<td>Format Option</td>
<td>Description</td>
<td>Syntax</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td><code>endquote</code></td>
<td>Data can be enclosed between two delimiters, specified with <code>quote</code> and <code>endquote</code>. The <code>quote</code> and <code>endquote</code> characters are removed during loading when specified. For example: <code>format =&gt; JSON_OBJECT('quote' value '(', 'endquote' value ')')</code></td>
<td><code>endquote:character</code></td>
</tr>
<tr>
<td><code>escape</code></td>
<td>The character <code>\</code> is used as the escape character when specified.</td>
<td><code>escape: true</code></td>
</tr>
<tr>
<td><code>ignoreblanklines</code></td>
<td>Blank lines are ignored when set to true.</td>
<td><code>ignoreblanklines: true</code></td>
</tr>
<tr>
<td><code>ignoremissingcolumns</code></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></td>
</tr>
<tr>
<td><code>jsonpath</code></td>
<td>JSON path to identify the document to load.</td>
<td><code>jsonpath: string</code></td>
</tr>
<tr>
<td><code>language</code></td>
<td>Specifies a language name (for example, FRENCH), from which locale-sensitive information can be derived.</td>
<td><code>language: string</code></td>
</tr>
<tr>
<td><code>maxdocsize</code></td>
<td>Maximum size of JSON documents.</td>
<td><code>maxdocsize: number</code></td>
</tr>
<tr>
<td><code>numericcharacters</code></td>
<td>Specifies the characters to use as the group separator and decimal character. <code>decimal_character</code>: The decimal separates the integer portion of a number from the decimal portion. <code>group_separator</code>: The group separator separates integer groups (that is, thousands, millions, billions, and so on).</td>
<td><code>numericcharacters: &quot;decimal_character group_separator&quot;</code></td>
</tr>
<tr>
<td><code>numberformat</code></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 <code>numericcharacters</code>.</td>
<td><code>numberformat: number_format_model</code></td>
</tr>
<tr>
<td>Format Option</td>
<td>Description</td>
<td>Syntax</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>partition_columns</td>
<td>The format option partition_columns is used with DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE to specify the column names and data types of partition columns when the partition columns are derived from the file path, depending on the type of data file, structured or unstructured:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• When the DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE includes the column_list parameter and the data file is unstructured, such as with CSV text files, partition_columns does not include the data type. For example, use a format such as the following for this type of partition_columns specification:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;partition_columns&quot;:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[&quot;state&quot;,&quot;zipcode&quot;]'</td>
</tr>
<tr>
<td></td>
<td>The data type is not required because it is specified in the DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE column_list parameter.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• When the DBMS_CLOUDCREATE_EXTERNAL_PART_TABLE does not include the column_list parameter and the data files are structured, such as Avro, ORC, or Parquet files, the partition_columns option includes the data type. For example, the following shows a partition_columns specification:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;partition_columns&quot;:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[{&quot;name&quot;:&quot;country&quot;,&quot;type&quot;:&quot;varchar2(10)&quot;},</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{&quot;name&quot;:&quot;year&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{&quot;name&quot;:&quot;month&quot;,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>{&quot;name&quot;:&quot;month&quot;,&quot;type&quot;:&quot;varchar2(10)&quot;}]'</td>
</tr>
<tr>
<td></td>
<td>If the data files are unstructured and the type sub-clause is specified with partition_columns, the type sub-clause is ignored.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For object names that are not based on hive format, the order of the partition_columns specified columns must match the order as they appear in the object name in the file_url_list.</td>
<td></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>Default value: Null meaning no quote</td>
<td></td>
</tr>
<tr>
<td>Format Option</td>
<td>Description</td>
<td>Syntax</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td><code>recorddelimiter</code></td>
<td>Specifies the record delimiter.</td>
<td><code>recorddelimiter: character</code></td>
</tr>
<tr>
<td>Option valid with JSON data</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 all files in the procedure. 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 &quot;\n&quot; as the record delimiter for all files in the procedure. Specify this argument explicitly if you want to override the default behavior, for example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>format =&gt; json_object('recorddelimiter' VALUE '''\r\n''')</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>To indicate that there is no record delimiter you can specify a <code>recorddelimiter</code> that does not occur in the input file. For example, to indicate that there is no delimiter, specify the control character 0x01 (SOH) as a value for the <code>recorddelimiter</code> and set the <code>recorddelimiter</code> value to &quot;0x'01'&quot; (this character does not occur in JSON text). For example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>format =&gt; '{&quot;recorddelimiter&quot; : &quot;0x'01'&quot;}'</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The <code>recorddelimiter</code> is set once per procedure call. If you are using the default value, detected newline, then all files use the same record delimiter, if one is detected.</td>
<td></td>
</tr>
<tr>
<td><code>rejectlimit</code></td>
<td>The operation will error out after specified number of rows are rejected.</td>
<td><code>rejectlimit: number</code></td>
</tr>
<tr>
<td>Option valid with JSON data</td>
<td>Default value: 0</td>
<td></td>
</tr>
<tr>
<td><code>removequotes</code></td>
<td>Removes any quotes that are around any field in the source file.</td>
<td><code>removequotes: true</code></td>
</tr>
<tr>
<td>Default value: False</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>skipheaders</code></td>
<td>Specifies how many rows should be skipped from the start of the file.</td>
<td><code>skipheaders: number</code></td>
</tr>
<tr>
<td>Default value: 0 if not specified, 1 if specified without a value</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>territory</code></td>
<td>Specifies a territory name to further determine input data characteristics.</td>
<td><code>territory: string</code></td>
</tr>
<tr>
<td>Default value: Null</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format Option</td>
<td>Description</td>
<td>Syntax</td>
</tr>
<tr>
<td>----------------</td>
<td>----------------------------------------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<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: string</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>The string can contain wildcard characters such as &quot;$&quot;.</td>
</tr>
<tr>
<td></td>
<td>MM/DD/YYYY HH:MI:SS.FF3</td>
<td></td>
</tr>
<tr>
<td>timestampltzformat</td>
<td>Specifies the timestamp with local timezone format in the source file. The format option AUTO searches for the following formats:</td>
<td>timestampltzformat: string</td>
</tr>
<tr>
<td></td>
<td>DD Mon YYYY HH:MI:SS.FF TZR</td>
<td>Default value: Database timestamp with local timezone format</td>
</tr>
<tr>
<td></td>
<td>MM/DD/YYYY HH:MI:SS.FF TZR</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 TZR</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: string</td>
</tr>
<tr>
<td></td>
<td>DD Mon YYYY HH:MI:SS.FF TZR</td>
<td>Default value: Database timestamp with timezone format</td>
</tr>
<tr>
<td></td>
<td>MM/DD/YYYY HH:MI:SS.FF TZR</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 TZR</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, then this option will truncate the value of the field rather than reject the row.</td>
<td>truncatecol: true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Default value: False</td>
</tr>
<tr>
<td>Format Option</td>
<td>Description</td>
<td>Syntax</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td><strong>type</strong></td>
<td>Specifies the source file type. See the description of CSV in field_definitions Clause. If the type is datapump, then the only other valid format option is rejectlimit. If the type is datapump, then the only Object Stores supported are Oracle Cloud Infrastructure Object Storage and Oracle Cloud Infrastructure Object Storage Classic. See DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet for type values avro, orc, or parquet. For JSON data with DBMS_CLOUD.COPY_COLLECTION type has two valid values: json (default) and ejson. For DBMS_CLOUD.COPY_COLLECTION these values both specify that the input is JSON data. The value ejson causes extended objects in the textual JSON input data to be translated to scalar JSON values in the native binary JSON collection. The value json does not perform this transformation and all objects in the input data are converted to binary JSON format. For JSON data with DBMS_CLOUD.COPY_DATA type has one valid value: json. This value specifies that the input is JSON data. • See COPY_COLLECTION Procedure for information on loading JSON objects. • See Objects That Extend JSON Scalars for information about extended JSON objects, which are interpreted when parameter type has value ejson.</td>
<td>type: csv</td>
</tr>
</tbody>
</table>

**unpackarrays**

Only use with COPY_COLLECTION

When set to true, if a loaded document is an array, then the contents of the array are loaded as documents rather than the array itself. This only applies to the top-level array. When set to true, the entire array is inserted as a single document. This option is valid only for JSON collection data with DBMS_CLOUD.COPY_COLLECTION.

unpackarrays: true

Default value: False
DBMS_CLOUD Package Format Options for EXPORT_DATA with Text Files (CSV, JSON, and XML)

Describes the valid format parameter options for DBMS_CLOUD.EXPORT_DATA with text file formats (CSV, JSON, and XML).

**Note:**
These are the valid format parameters for use with DBMS_CLOUD.EXPORT_DATA for use with text file output (with format type values csv, json, or xml). See EXPORT_DATA Procedure for the format options when the format type is datapump.

The two ways to specify the format argument are:

format => '{"format_option" : "format_value" }'

And:

format => json_object('format_option' value 'format_value')

Examples:

format => json_object('type' VALUE 'json')

To specify multiple format options, separate the values with a ",, .

For example:

format => json_object('compression' value 'gzip', 'type' value 'json')

This table covers the format options for DBMS_CLOUD.EXPORT_DATA when the output file is CSV, JSON, or XML. For other procedures and other output types, see DBMS_CLOUD Package Format Options for the list of format options.

<table>
<thead>
<tr>
<th>Format Option</th>
<th>Description</th>
<th>Syntax</th>
<th></th>
</tr>
</thead>
</table>
| compression   | Specifies the compression type of the source file. Note: ZIP archiving format is not supported. | compression: gzip | Default value: Null value meaning no compression.
| delimiter     | Specifies a custom field delimiter. This option only applies with csv type. format => json_object('delimiter' value 'value') | delimiter: character | Default value , (comma)The delimiter value cannot be an ASCII code or an escape character.
<table>
<thead>
<tr>
<th>Format Option</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| **endquote** | Specifies that fields can be enclosed between two delimiters, with *quote* and *endquote*. If *endquote* is not specified, then the *quote* character will be used by default as the *endquote* character. This option only applies with *csv* type. For example: | endquote: *character*
| | | Default value: Null, meaning no *endquote*.
| **escape** | Specifies the occurrence of *quote* character in the field value using "\" character. This option only applies with *csv* type. | escape: *true*|
| | | Default value: *false*.
| **header** | Writes column names as the first line in output files of *csv* type. The *header* option can accept a boolean or a string value. The valid values are:
| &bull; false: Skips the header row.
| &bull; true: Includes the header row. The column names are based on the *SELECT* statement in the *query* parameter. You must specify column aliases in the *SELECT* statement when using virtual columns or expressions.
| &bull; String to define custom header names: Enables you to define header rows with custom names. The number of columns and delimiters in the string value must match the number of columns and delimiters in the *SELECT* statement. The default delimiter is comma (,). The *header* option only applies with *csv* type. | header: *true| *false| String to define custom header names*|
| | | Default value: *false*.
<p>| <strong>fileextension</strong> | Custom file extension to override the default choice for the format type. This applies to text formats with <code>DBMS_CLOUD.EXPORT_DATA</code>: CSV, JSON, and XML. If the specified string does not start with period (dot), then a dot is automatically inserted before the file extension in the final file name. If no file extension is desired, use the value: | Valid values: Any file extension. Default value: Depends on the format type option: • CSV format: <code>.csv</code> • JSON format: <code>.json</code> • XML format: <code>.xml</code> |
| | | Minimum value: 10485760 (10MB) Maximum value: 2GB Default value: 10485760 (10MB) |
| <strong>maxfilesize</strong> | Number in bytes for maximum size of output generated. This applies to text based formats for exporting data with <code>DBMS_CLOUD.EXPORT_DATA</code> when the format type option is set to <em>csv</em>, <em>json</em>, or <em>xml</em>. |</p>
<table>
<thead>
<tr>
<th>Format Option</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
</table>
| quote         | In CSV format, fields can be enclosed between two delimiters. Specify the delimiters with `quote` and `endquote`. If `endquote` is not specified, then the quote character will be used by default as the endquote character. This option only applies with `csv` type. | `quote: character`  
Default value: Null meaning do not enclose fields with quotes.                        |
| trimspaces    | Specifies how the leading and trailing spaces of the fields are trimmed for CSV format. Trim spaces is applied before quoting the field, if the `quote` parameter is specified. This option only applies with `csv` type. See the description of `trim_spec`. | `trimspaces: rtrim| ltrim| notrim| lrtrim| ldrtrim`  
Default value: `notrim`                                                     |
| type          | Specifies the output file type. `csv`: specifies Character Separated Values (CSV) format that allows you to export query results as a set of column values separated by any custom character.  
`json`: specifies to export the query results as JSON files.  
`xml`: specifies to export query results as rows of valid XML documents. Each row is encapsulated in a root XML tag of `<RECORD> </RECORD>`. The query result is automatically transformed into XML format using `XMLFOREST` SQL function. Use Column Aliases to customize the XML tag names for columns. | `type: csv|json|xml` |

### DBMS_CLOUD Avro, ORC, and Parquet Support

This section covers the DBMS_CLOUD Avro, ORC, and Parquet support provided with Autonomous Database.

#### Topics
- DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet
- DBMS_CLOUD Package Avro to Oracle Data Type Mapping
- DBMS_CLOUD Package ORC to Oracle Data Type Mapping
- DBMS_CLOUD Package Parquet to Oracle Data Type Mapping
- DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types
- DBMS_CLOUD Package Avro, ORC, and Parquet to Oracle Column Name Mapping

### DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet

The format argument in DBMS_CLOUD specifies the format of source files.

The two ways to specify the format argument are:

format => '{"format_option" : "format_value"}'
And:

```json
format => json_object('format_option' value 'format_value'))
```

Examples:

```json
format => json_object('type' VALUE 'CSV')
```

To specify multiple format options, separate the values with a ",".

For example:

```json
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 : avro</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 Avro, ORC, or Parquet file metadata. The column names will match those found in Avro, ORC, or Parquet. The data types are converted from Avro, ORC, or Parquet data types to Oracle data types. All columns are added to the table. The value first specifies to use the metadata from the first Avro, ORC, or Parquet file in the file_uri_list to auto generate the columns and their data types. Use first if all of the files have the same schema. The value all specifies to use the metadata from all Avro, ORC, or Parquet files in the file_uri_list to auto generate the columns and their data types. Use all (slower) if the files may have different schemas. <strong>Default:</strong> If column_list is specified, then the schema value, if specified is ignored. If column_list is not specified then the schema default value is first. <strong>Note:</strong> For Avro, ORC, or Parquet format files the schema format option is not available and the column_list parameter must be specified for partitioned external tables using the DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE procedure.</td>
<td>schema : first</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:**

Complex types, such as maps, arrays, and structs are supported starting with Oracle Database 19c. See DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types for information on using Avro complex types.
### Avro Type

<table>
<thead>
<tr>
<th>Avro Type</th>
<th>Oracle Type</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>NUMBER(10)</td>
<td></td>
</tr>
<tr>
<td>LONG</td>
<td>NUMBER(19)</td>
<td></td>
</tr>
<tr>
<td>BOOL</td>
<td>NUMBER(1)</td>
<td></td>
</tr>
<tr>
<td>UTF8_BYTE_ARRAY</td>
<td>RAW(2000)</td>
<td></td>
</tr>
<tr>
<td>FLT</td>
<td>BINARY_FLOAT</td>
<td></td>
</tr>
<tr>
<td>DBL</td>
<td>BINARY_DOUBLE</td>
<td></td>
</tr>
<tr>
<td>DECIMAL(p)</td>
<td>NUMBER(p)</td>
<td></td>
</tr>
<tr>
<td>DECIMAL(p,s)</td>
<td>NUMBER(p,s)</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>STRING</td>
<td>VARCHAR2</td>
<td></td>
</tr>
<tr>
<td>TIME_MILLIS</td>
<td>VARCHAR2(20 BYTE)</td>
<td></td>
</tr>
<tr>
<td>TIME_MICROS</td>
<td>VARCHAR2(20 BYTE)</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP_MILLIS</td>
<td>TIMESTAMP(3)</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP_MICROS</td>
<td>TIMESTAMP(6)</td>
<td></td>
</tr>
<tr>
<td>ENUM</td>
<td>VARCHAR2(n)</td>
<td>Where: &quot;n&quot; is the actual maximum length of the AVRO ENUM's possible values</td>
</tr>
<tr>
<td>DURATION</td>
<td>RAW(2000)</td>
<td></td>
</tr>
<tr>
<td>FIXED</td>
<td>RAW(2000)</td>
<td></td>
</tr>
<tr>
<td>NULL</td>
<td>VARCHAR2(1) BYTE</td>
<td></td>
</tr>
</tbody>
</table>

See [DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types](#) for information on using Avro complex types.

### DBMS_CLOUD Package ORC to Oracle Data Type Mapping

Describes the mapping of ORC data types to Oracle data types.

See [DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types](#) for information on using ORC complex types.

<table>
<thead>
<tr>
<th>ORC Type</th>
<th>Oracle Type</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>array</td>
<td>VARCHAR2(n) JSON format</td>
<td>DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types</td>
</tr>
<tr>
<td>bigint (64 bit)</td>
<td>NUMBER(19)</td>
<td></td>
</tr>
<tr>
<td>binary</td>
<td>BLOB</td>
<td></td>
</tr>
<tr>
<td>boolean (1 bit)</td>
<td>NUMBER(1)</td>
<td></td>
</tr>
<tr>
<td>char</td>
<td>CHAR(n)</td>
<td></td>
</tr>
<tr>
<td>date</td>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>BINARY_DOUBLE</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>BINARY_FLOAT</td>
<td></td>
</tr>
<tr>
<td>int (32 bit)</td>
<td>NUMBER(10)</td>
<td></td>
</tr>
<tr>
<td>list</td>
<td>VARCHAR2(n) JSON format</td>
<td>DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types</td>
</tr>
</tbody>
</table>

See [DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types](#) for information on using ORC complex types.
### DBMS_CLOUD Package Parquet to Oracle Data Type Mapping

Describes the mapping of Parquet data types to Oracle data types.

**Note:**

Complex types, such as maps, arrays, and structs are supported starting with Oracle Database 19c. See [DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types](#) for information on using Parquet complex types.

<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>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_MILLIS_UTC</td>
<td>VARCHAR2(20 BYTE)</td>
</tr>
<tr>
<td>TIME_MICROS</td>
<td>VARCHAR2(20 BYTE)</td>
</tr>
</tbody>
</table>
DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types

Describes the mapping of Avro, ORC, and Parquet complex data types to Oracle data types. Autonomous Database supports complex data types, including the following complex types:

- struct
- list
- map
- union
- array

When you specify a source file type of Avro, ORC, or Parquet and the source file includes complex columns, Autonomous Database queries return JSON for the complex columns. This simplifies processing of query results; you can use Oracle's powerful JSON parsing features consistently across the file types and data types. The following table shows the format for the complex types in Autonomous Database:

<table>
<thead>
<tr>
<th>Type</th>
<th>Parquet</th>
<th>ORC</th>
<th>Avro</th>
<th>Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>List: sequence of</td>
<td>List</td>
<td>List</td>
<td>Array</td>
<td>VARCHAR2 (JSON</td>
</tr>
<tr>
<td>values</td>
<td></td>
<td></td>
<td></td>
<td>format)</td>
</tr>
<tr>
<td>Map: list of objects with single key</td>
<td>Map</td>
<td>Map</td>
<td>Map</td>
<td>VARCHAR2 (JSON format)</td>
</tr>
<tr>
<td>Union: values of different type</td>
<td>Not Available</td>
<td>Union</td>
<td>Union</td>
<td>VARCHAR2 (JSON format)</td>
</tr>
<tr>
<td>Object: zero or more key-value pairs</td>
<td>Struct</td>
<td>Struct</td>
<td>Record</td>
<td>VARCHAR2 (JSON format)</td>
</tr>
</tbody>
</table>

Note:

The complex fields map to VARCHAR2 columns and VARCHAR2 size limits apply.

If your ORC, Parquet, or Avro source files contain complex types, then you can query the JSON output for these common complex types. For example, the following shows an ORC file, movie-info.orc, with a complex type (the same complex type handling applies for Parquet and Avro source files).
Consider the *movie-info.orc* file with the following schema:

```sql
id    int
original_title string
overview       string
poster_path    string
release_date   string
vote_count     int
runtime        int
popularity     double
genres        array<struct{id:int,name:string}>
```

Notice that each movie is categorized by multiple *genres* using an array of *genres*. The *genres* array is an array of structs and each item has an *id* (int) and a *name* (string). The *genres* array is considered a complex type. You can create a table over this ORC file using `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` as follows:

```
BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
    table_name =>'movie_info',
    credential_name =>'OBJ_STORE_CRED',
    file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/n/mytenancy/b/movies/o/movie-info.orc',
    format => '{"type":"orc", "schema": "first"}');
END;
/
```

When you create the external table the database automatically generates the columns based on the schema in the ORC file (if you are using Avro or Parquet, the same applies). For this example, the `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` creates a table in your database as follows:

```
CREATE TABLE "ADMIN"."MOVIE_INFO"  
( "ID" NUMBER(10,0),
  "ORIGINAL_TITLE" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
  "OVERVIEW" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
  "POSTER_PATH" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
  "RELEASE_DATE" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
  "VOTE_COUNT" NUMBER(10,0),
  "RUNTIME" NUMBER(10,0),
  "POPULARITY" BINARY_DOUBLE,
  "GENRES" VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP"
)  DEFAULT COLLATION "USING_NLS_COMP"  
ORGANIZATION EXTERNAL  
( TYPE      ORACLE_BIGDATA  
DEFAULT DIRECTORY "DATA_PUMP_DIR"  
ACCESS PARAMETERS  
  ( com.oracle.bigdata.credential.name=OBJ_STORE_CRED  
    com.oracle.bigdata.fileformat=ORC  
  )  
)  
LOCATION  
{  
```

Appendix A
DBMS_CLOUD Package

A-74
Now you can query the movie data:

```sql
SELECT original_title, release_date, genres
FROM movie_info
WHERE release_date > '2000'
ORDER BY original_title;
```

This produces the following output:

<table>
<thead>
<tr>
<th>original_title</th>
<th>release_date</th>
<th>genres</th>
</tr>
</thead>
<tbody>
<tr>
<td>(500) Days of Summer</td>
<td>2009</td>
<td>[&quot;id&quot;:3,&quot;name&quot;:&quot;Drama&quot;], [&quot;id&quot;:6,&quot;name&quot;:&quot;Comedy&quot;], [&quot;id&quot;:17,&quot;name&quot;:&quot;Horror&quot;], [&quot;id&quot;:19,&quot;name&quot;:&quot;Western&quot;], [&quot;id&quot;:18,&quot;name&quot;:&quot;War&quot;], [&quot;id&quot;:15,&quot;name&quot;:&quot;Romance&quot;]</td>
</tr>
<tr>
<td>10,000 BC</td>
<td>2008</td>
<td>[&quot;id&quot;:6,&quot;name&quot;:&quot;Comedy&quot;]</td>
</tr>
<tr>
<td>11:14</td>
<td>2003</td>
<td>[&quot;id&quot;:9,&quot;name&quot;:&quot;Thriller&quot;], [&quot;id&quot;:14,&quot;name&quot;:&quot;Family&quot;]</td>
</tr>
<tr>
<td>127 Hours</td>
<td>2010</td>
<td>[&quot;id&quot;:6,&quot;name&quot;:&quot;Comedy&quot;], [&quot;id&quot;:3,&quot;name&quot;:&quot;Drama&quot;]</td>
</tr>
<tr>
<td>13 Going on 30</td>
<td>2004</td>
<td>[&quot;id&quot;:6,&quot;name&quot;:&quot;Comedy&quot;], [&quot;id&quot;:3,&quot;name&quot;:&quot;Drama&quot;], [&quot;id&quot;:18,&quot;name&quot;:&quot;War&quot;], [&quot;id&quot;:15,&quot;name&quot;:&quot;Romance&quot;]</td>
</tr>
<tr>
<td>1408</td>
<td>2007</td>
<td>[&quot;id&quot;:45,&quot;name&quot;:&quot;Sci-Fi&quot;], [&quot;id&quot;:6,&quot;name&quot;:&quot;Comedy&quot;], [&quot;id&quot;:17,&quot;name&quot;:&quot;Horror&quot;], [&quot;id&quot;:6,&quot;name&quot;:&quot;Comedy&quot;], [&quot;id&quot;:18,&quot;name&quot;:&quot;War&quot;]</td>
</tr>
</tbody>
</table>

Notice that the complex type `genres` is returned as a JSON array.

To make the JSON data more useful, you can transform the column using Oracle’s JSON functions. For example, you can use the JSON "." notation as well as the more powerful transform functions such as `JSON_TABLE`.

See Simple Dot-Notation Access to JSON Data for information on "." notation.

See SQL/JSON Function `JSON_TABLE` for information on `JSON_TABLE`.

The following example shows a query on the table that takes each value of the array and turns the value into a row in the result set:

```sql
SELECT original_title, release_date, m.genre_name, genres
FROM movie_info mi,
     JSON_TABLE(mi.genres, '$.name[*]' COLUMNS (genre_name VARCHAR2(25) PATH '$') ) AS m
WHERE rownum < 10;
```
Appendix A

DBMS_CLOUD Package

The JSON_TABLE creates a row for each value of the array, think outer join, and the
struct is parsed to extract the name of the genre. This produces the following output:

original_title
release_date
genre_name
genres
(500) Days of Summer
2009
Drama
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},
{"id":17,"name":"Horror"},{"id":19,"name":"Western"},
{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer
2009
Comedy
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},
{"id":17,"name":"Horror"},{"id":19,"name":"Western"},
{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer
2009
Horror
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},
{"id":17,"name":"Horror"},{"id":19,"name":"Western"},
{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer
2009
Western
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},
{"id":17,"name":"Horror"},{"id":19,"name":"Western"},
{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer
2009
War
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},
{"id":17,"name":"Horror"},{"id":19,"name":"Western"},
{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer
2009
Romance
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},
{"id":17,"name":"Horror"},{"id":19,"name":"Western"},
{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
10,000 BC
2008
Comedy
[{"id":6,"name":"Comedy"}]
11:14
2003
Family
[{"id":9,"name":"Thriller"},
{"id":14,"name":"Family"}]
11:14
2003
Thriller
[{"id":9,"name":"Thriller"},
{"id":14,"name":"Family"}]
127 Hours
2010
Comedy
[{"id":6,"name":"Comedy"},{"id":3,"name":"Drama"}]
127 Hours
2010
Drama
[{"id":6,"name":"Comedy"},{"id":3,"name":"Drama"}]
13 Going on 30
2004

A-76


DBMS_CLOUD Package Avro, ORC, and Parquet to Oracle Column Name Mapping

Describes rules for how Avro, ORC, and Parquet column names are converted to Oracle column names.

The following are supported for Avro, ORC, and Parquet column names, but may require use of double quotes for Oracle SQL references in external tables. Thus, for ease of use and to avoid having to use double quotes when referencing column names, if possible do not use the following in Avro, ORC, and Parquet column names:

- Embedded blanks
- Leading numbers
- Leading underscores
- Oracle SQL reserved words

The following table shows various types of Avro, ORC, and Parquet column names, and rules for using the column names in Oracle column names in external tables.

<table>
<thead>
<tr>
<th>Avro, ORC, or Parquet Name</th>
<th>CREATE TABLE Name</th>
<th>Oracle CATALOG Name</th>
<th>Valid SQL</th>
<th>Notes</th>
</tr>
</thead>
</table>
| part, Part, or PART        | part, Part, PART  | PART                | select part  
select Part  
select paRt  
select PART | Oracle implicitly uppercases unquoted column names |
<p>| <strong>index_key</strong>              | <strong>index_key</strong>     | <strong>index_key</strong>       | select &quot;<strong>index_key</strong>&quot; | Double quotes are required when there is a leading underscore, which also preserves the character case |</p>
<table>
<thead>
<tr>
<th>Avro, ORC, or Parquet Name</th>
<th>CREATE TABLE Name</th>
<th>Oracle CATALOG</th>
<th>Valid SQL</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<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, ORC, or Parquet column name.
- After using `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` to create an external table with the format specified as `avro`, `orc`, or `parquet`, use the `DESCRIBE` command in SQL*Plus to view the table’s column names.
- When Oracle SQL Reserved Words are used in Avro, ORC, or Parquet column names, they must always be double-quoted when referenced anywhere in SQL. See Oracle SQL Reserved Words for more information.

DBMS_CLOUD Exceptions

The following table describes exceptions for DBMS_CLOUD.

<table>
<thead>
<tr>
<th>Exception</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reject_limit</td>
<td>20003</td>
<td>The reject limit of an external table was reached.</td>
</tr>
<tr>
<td>credential_not_exist</td>
<td>20004</td>
<td>A credential object does not exist.</td>
</tr>
<tr>
<td>table_not_exist</td>
<td>20005</td>
<td>A table does not exist.</td>
</tr>
<tr>
<td>unsupported_obj_store</td>
<td>20006</td>
<td>An unsupported object store URI was provided.</td>
</tr>
<tr>
<td>iden_too_long</td>
<td>20008</td>
<td>An identifier is too long.</td>
</tr>
<tr>
<td>invalid_format</td>
<td>20009</td>
<td>A format argument is not valid.</td>
</tr>
<tr>
<td>missing_credential</td>
<td>20010</td>
<td>Mandatory credential object information was not specified.</td>
</tr>
<tr>
<td>invalid_object_uri</td>
<td>20011</td>
<td>An invalid object URI was provided.</td>
</tr>
<tr>
<td>invalid_partitioning_clause</td>
<td>20012</td>
<td>An partitioning clause is missing or was not provided.</td>
</tr>
<tr>
<td>unsupported_feature</td>
<td>20013</td>
<td>An unsupported feature was used that is not existent in the current database version.</td>
</tr>
<tr>
<td>part_not_exist</td>
<td>20014</td>
<td>A partition or subpartition does not exist, or a table is not a partitioned external table or hybrid partitioned table.</td>
</tr>
<tr>
<td>invalid_table_name</td>
<td>20016</td>
<td>An invalid table name was used.</td>
</tr>
<tr>
<td>invalid_schema_name</td>
<td>20017</td>
<td>An invalid schema name was used.</td>
</tr>
<tr>
<td>invalid_dir_name</td>
<td>20018</td>
<td>An invalid directory name was used.</td>
</tr>
<tr>
<td>invalid_file_name</td>
<td>20019</td>
<td>An invalid file name was used.</td>
</tr>
<tr>
<td>invalid_cred_attribute</td>
<td>20020</td>
<td>Invalid credential attributes were specified.</td>
</tr>
<tr>
<td>table_exist</td>
<td>20021</td>
<td>A table already exists.</td>
</tr>
<tr>
<td>credential_exist</td>
<td>20022</td>
<td>A credential object already exists.</td>
</tr>
<tr>
<td>invalid_req_method</td>
<td>20023</td>
<td>A request method is either too long or invalid.</td>
</tr>
<tr>
<td>invalid_req_header</td>
<td>20024</td>
<td>An invalid request header was specified.</td>
</tr>
</tbody>
</table>
### DBMS_CLOUD_ADMIN Package

The **DBMS_CLOUD_ADMIN** package provides administrative routines for configuring a database.

**Topics**
- Summary of DBMS_CLOUD_ADMIN Subprograms
- DBMS_CLOUD_ADMIN Exceptions

### Summary of DBMS_CLOUD_ADMIN Subprograms

This section covers the **DBMS_CLOUD_ADMIN** subprograms provided with Autonomous Database.

<table>
<thead>
<tr>
<th>Subprogram</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE_DATABASE_LINK Procedure</td>
<td>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.</td>
</tr>
<tr>
<td>DISABLE_APP_CONT Procedure</td>
<td>This procedure disables database application continuity for the session associated with the specified service name in Autonomous Database.</td>
</tr>
<tr>
<td>DISABLE_EXTERNAL_AUTHENTICATION Procedure</td>
<td>This procedure disables external authentication for the Autonomous Database instance.</td>
</tr>
<tr>
<td>DISABLE_PRINCIPAL_AUTH Procedure</td>
<td>This procedure revokes principal based authentication for the specified provider and applies to the ADMIN user or to the specified user.</td>
</tr>
<tr>
<td>DISABLE_RESOURCE_PRINCIPAL Procedure</td>
<td>This procedure disables resource principal credential and creates the credential OCI$RESOURCE_PRINCIPAL. With a user name specified, other than ADMIN, the procedure grants the specified schema access to the resource principal credential.</td>
</tr>
<tr>
<td>DROP_DATABASE_LINK Procedure</td>
<td>This procedure drops a database link.</td>
</tr>
<tr>
<td>ENABLE_APP_CONT Procedure</td>
<td>This procedure enables database application continuity for the session associated with the specified service name in Autonomous Database.</td>
</tr>
<tr>
<td>ENABLE_AWS_Arn Procedure</td>
<td>This procedure enables a user to create AWS ARN credentials in Autonomous Database.</td>
</tr>
<tr>
<td>Subprogram</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENABLE_EXTERNAL_AUTHENTICATION Procedure</td>
<td>This procedure enables a user to logon to Autonomous Database using the specified external authentication scheme.</td>
</tr>
<tr>
<td>ENABLE_FEATURE Procedure</td>
<td>This procedure enables the specified feature on the Autonomous Database instance.</td>
</tr>
<tr>
<td>ENABLE_PRINCIPAL_AUTH Procedure</td>
<td>This procedure enables principal authentication for the specified provider and applies to the ADMIN user or the specified user.</td>
</tr>
<tr>
<td>ENABLE_RESOURCE_PRINCIPAL Procedure</td>
<td>This procedure enables resource principal credential and creates the credential OCI$RESOURCE_PRINCIPAL. With a user name specified, other than ADMIN, the procedure grants the specified schema access to the resource principal credential.</td>
</tr>
<tr>
<td>REPLAY_WORKLOAD Procedure</td>
<td>This procedure is overloaded. It initiates the workload replay.</td>
</tr>
</tbody>
</table>

**CREATE_DATABASE_LINK Procedure**

This procedure creates a database link to a target database in the schema calling the API. The overloaded form enables you to create a database link with Oracle-managed heterogeneous connectivity to a supported non-Oracle database.

**Syntax**

```
DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
   db_link_name         IN VARCHAR2,
   hostname             IN VARCHAR2,
   port                 IN NUMBER,
   service_name         IN VARCHAR2,
   ssl_server_cert_dn   IN VARCHAR2,
   credential_name      IN VARCHAR2,
   directory_name       IN VARCHAR2 DEFAULT,
   gateway_link         IN BOOLEAN DEFAULT,
   public_link          IN BOOLEAN DEFAULT,
   private_target       IN BOOLEAN DEFAULT);
```

```
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,
   credential_name      IN VARCHAR2,
   directory_name       IN VARCHAR2 DEFAULT,
   gateway_params       IN CLOB 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></td>
<td>When you specify a connection with Oracle-managed heterogeneous connectivity by supplying the <code>gateway_params</code> parameter, note the following:</td>
</tr>
<tr>
<td></td>
<td>• When the <code>db_type</code> value is <code>SNOWFLAKE</code> the <code>hostname</code> is the Snowflake account identifier. To find your Snowflake account identifier, see <a href="#">Account Identifier Formats by Cloud Platform and Region</a>.</td>
</tr>
<tr>
<td>port</td>
<td>The port for the connections to the target database.</td>
</tr>
<tr>
<td></td>
<td>To ensure security, when the connection is to an Oracle Database, ports are restricted to: 1521-1525, or 2484.</td>
</tr>
<tr>
<td></td>
<td>When you specify a connection with Oracle-managed heterogeneous connectivity using the <code>gateway_params</code> parameter, set the port based on the <code>db_type</code> value:</td>
</tr>
<tr>
<td></td>
<td>• AWSREDSHIFT: use port 5439</td>
</tr>
<tr>
<td></td>
<td>• MYSQL: use port 3306</td>
</tr>
<tr>
<td></td>
<td>• POSTGRES: use port 5432</td>
</tr>
<tr>
<td></td>
<td>• SNOWFLAKE: use port 443</td>
</tr>
<tr>
<td></td>
<td>See <a href="#">Oracle-Managed Heterogeneous Connectivity Database Types and Ports</a> for more information.</td>
</tr>
<tr>
<td>service_name</td>
<td>The <code>service_name</code> for the database to link to. For a target Autonomous Database, find the service name by one of the following methods:</td>
</tr>
<tr>
<td></td>
<td>• Look in the <code>tnsnames.ora</code> file in the <code>wallet.zip</code> that you download from an Autonomous Database for your connection.</td>
</tr>
<tr>
<td></td>
<td>• Click <strong>DB Connection</strong> on the Oracle Cloud Infrastructure Console. In the <strong>Connection Strings</strong> area, each connection string includes a <code>service_name</code> entry with the connection string for the corresponding service. When both Mutual TLS (mTLS) and TLS connections are allowed, under <strong>TLS Authentication</strong> select <strong>TLS</strong> to view the TNS names and connection strings for connections with TLS authentication. See <a href="#">View TNS Names and Connection Strings for an Autonomous Database Instance</a> for more information.</td>
</tr>
<tr>
<td></td>
<td>• Query <code>V$SERVICES</code> view. For example:</td>
</tr>
<tr>
<td></td>
<td>```</td>
</tr>
<tr>
<td></td>
<td>SELECT name FROM V$SERVICES;</td>
</tr>
<tr>
<td></td>
<td>```</td>
</tr>
<tr>
<td></td>
<td>When you specify a connection with Oracle-managed heterogeneous connectivity using the <code>gateway_params</code> parameter, the <code>service_name</code> is the <code>database_name</code> of the non-Oracle database.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ssl_server_cert_dn</td>
<td>The DN value found in the server certificate. Oracle-managed heterogeneous connectivity is preconfigured with a wallet that contains most of the common trusted root and intermediate SSL certificates. The ssl_server_cert_dn must be NULL when you supply the gateway_params parameter. To connect to an Oracle Database on a private endpoint without a wallet, specify a NULL value for the ssl_server_cert_dn parameter. In addition, to connect to an Oracle Database with TCP, the following must be true:  - The target database must be on a private endpoint.  - The directory_name parameter must be NULL.  - The private_target parameter must be TRUE.</td>
</tr>
<tr>
<td>credential_name</td>
<td>The name of a stored credential created with DBMS_CLOUD.CREATE_CREDENTIAL. This is the credentials to access the target database.</td>
</tr>
<tr>
<td>directory_name</td>
<td>The directory for the stored cwallet.sso file. The default value for this parameter is 'data_pump_dir'. Oracle-managed heterogeneous connectivity is preconfigured with a wallet that contains most of the common trusted root and intermediate SSL certificates. The directory_name is not required when you supply the gateway_params parameter. To connect to an Oracle Database with TCP, specify a NULL value for the directory_name parameter. In addition, to connect to an Oracle Database with TCP, the following must be true:  - The target database must be on a private endpoint.  - The directory_name parameter must be NULL.  - The private_target parameter must be TRUE.</td>
</tr>
<tr>
<td>gateway_link</td>
<td>Indicates if the database link is created to another Oracle Database or to an Oracle Database Gateway. If gateway_link is set to FALSE, this specifies a database link to another Autonomous Database or to another Oracle Database. If gateway_link is set to TRUE, this specifies a database link to a non-Oracle system. This creates a connect descriptor in the database link definition that specifies (HS=OK). The default value for this parameter is FALSE.</td>
</tr>
<tr>
<td>public_link</td>
<td>Indicates if the database link is created as a public database link. To run DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK with this parameter set to TRUE, the user invoking the procedure must have EXECUTE privilege on the credential associated with the public database link and must have the CREATE PUBLIC DATABASE LINK system privilege. The EXECUTE privilege on the credential can be granted either by the ADMIN user or by the credential owner. The default value for this parameter is FALSE.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>private_target</td>
<td>When a database link accesses a hostname that needs to be resolved in a VCN DNS server, specify the private_target parameter with value TRUE.</td>
</tr>
<tr>
<td></td>
<td>When private_target is TRUE, the hostname parameter must be a single hostname. Using an IP address or SCAN IP for the hostname is not supported.</td>
</tr>
<tr>
<td></td>
<td>The default value for this parameter is FALSE.</td>
</tr>
<tr>
<td>gateway_params</td>
<td>Specify the target database type for Oracle-managed heterogeneous connectivity to connect to non-Oracle databases. The db_type value must be one of:</td>
</tr>
<tr>
<td></td>
<td>• AWSREDSHIFT</td>
</tr>
<tr>
<td></td>
<td>• AZURE</td>
</tr>
<tr>
<td></td>
<td>• MYSQL</td>
</tr>
<tr>
<td></td>
<td>• POSTGRES</td>
</tr>
<tr>
<td></td>
<td>• SNOWFLAKE</td>
</tr>
<tr>
<td></td>
<td>Specify the parameter with the json_object form. For example:</td>
</tr>
</tbody>
</table>
|              | gateway_params => json_object('db_type' value
|              | 'AWSREDSHIFT')                                                                                                                              |
|              | See Oracle-Managed Heterogeneous Connectivity Database Types and Ports for required port values for each database type.                                                                               |

### Usage Notes

- When you use the private_target parameter, note that database links from an Autonomous Database to a database service that is on a private endpoint are only supported in commercial regions and US Government regions.

This feature is enabled by default in all commercial regions.

This feature is enabled by default in US Government regions for newly provisioned databases.

For existing US Government databases on a private endpoint, if you want to create database links from an Autonomous Database to a target in a US Government region, please file a Service Request at Oracle Cloud Support and request to enable the private endpoint in government regions database linking feature.

US Government regions include the following:
- Oracle Cloud Infrastructure US Government Cloud with FedRAMP Authorization
- Oracle Cloud Infrastructure US Federal Cloud with DISA Impact Level 5 Authorization

- When connecting to a non-Oracle database, database linking is only supported if the target database is accessible through a public IP or a public hostname. See Create Database Links to Non-Oracle Databases for more information.

- To run DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK with a user other than ADMIN you need to grant EXECUTE and CREATE DATABASE LINK privileges to
that user. For example, run the following command as ADMIN to grant privileges to adb_user:

```
GRANT EXECUTE ON DBMS_CLOUD_ADMIN TO adb_user;
GRANT CREATE DATABASE LINK TO adb_user;
```

- Only one wallet file is valid per directory specified with the `directory_name` parameter. You can only upload one `cwallet.sso` at a time to the directory you choose for wallet files. This means with a `cwallet.sso` in a directory, you can only create database links to the databases for which the wallet in that directory is valid. To use multiple `cwallet.sso` files with database links you need to create additional directories and put each `cwallet.sso` in a different directory.

See [Create Directory in Autonomous Database](#) for information on creating directories.

- To create a database link to an Autonomous Database, set `GLOBAL_NAMES` to `FALSE` on the source database (non-Autonomous Database).

```
SQL> ALTER SYSTEM SET GLOBAL_NAMES = FALSE;
System altered.
```

```
SQL> SHOW PARAMETER GLOBAL_NAMES
NAME                     TYPE        VALUE
----------------------   -----------  -----------
global_names             boolean     FALSE
```

- When the `private_target` parameter is `TRUE`, the `hostname` parameter specifies a private host inside the VCN.

### Examples

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
      credential_name => 'DB_LINK_CRED',
      username => 'adb_user',
      password => 'password');
  DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
      db_link_name => 'SALESLINK',
      hostname => 'adb.eu-frankfurt-1.oraclecloud.com',
      port => '1522',
      service_name => 'example_medium.adb.example.oraclecloud.com',
      ssl_server_cert_dn => 'CN=adb.example.oraclecloud.com,OU=Oracle BMCS FRANKFURT,O=Oracle Corporation,L=Redwood City,ST=California,C=US',
      credential_name => 'DB_LINK_CRED');
END;
/
```

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
      credential_name => 'AWS_REDSHIFT_LINK_CRED',
      username => 'NICK',
      password => 'password');
};
```
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.</td>
</tr>
<tr>
<td></td>
<td>To find service names:</td>
</tr>
<tr>
<td></td>
<td>• Look in the tnsnames.ora file in the wallet.zip that you download from an</td>
</tr>
<tr>
<td></td>
<td>Autonomous Database for your connection.</td>
</tr>
<tr>
<td></td>
<td>• Click DB Connection on the Oracle Cloud Infrastructure Console. In the</td>
</tr>
<tr>
<td></td>
<td>Connection Strings area, each connection string includes a service_name entry</td>
</tr>
<tr>
<td></td>
<td>that contains the connection string for the corresponding service. When both</td>
</tr>
<tr>
<td></td>
<td>TLS Authentication select TLS to view the TNS names and connection strings for</td>
</tr>
<tr>
<td></td>
<td>connections with TLS authentication. See View TNS Names and Connection Strings</td>
</tr>
<tr>
<td></td>
<td>for an Autonomous Database Instance for more information.</td>
</tr>
<tr>
<td></td>
<td>• Query V$SERVICES view. For example:</td>
</tr>
<tr>
<td></td>
<td>SELECT name FROM V$SERVICES;</td>
</tr>
</tbody>
</table>

Usage Notes

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

Example

```sql
BEGIN
   DBMS_CLOUD_ADMIN.DISABLE_APP_CONT(
       service_name => 'nv123abc1_adb1_high.adb.oraclecloud.com'
   );
END;
/
```

Verify the value as follows:

```sql
SELECT name, failover_type FROM DBA_SERVICES;
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>FAILOVER_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>nv123abcl_adbl_high.adb.oraclecloud.com</td>
<td></td>
</tr>
</tbody>
</table>

DISABLE_EXTERNAL_AUTHENTICATION Procedure

Disables user authentication with external authentication schemes for the database.

Syntax

```sql
DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
```
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid_ext_auth</td>
<td>ORA-20004</td>
<td>See the accompanying message for a detailed explanation.</td>
</tr>
</tbody>
</table>

Example

BEGIN
    DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
END;
/

PL/SQL procedure successfully completed.

DISABLE_PRINCIPAL_AUTH Procedure

This procedure revokes principal based authentication for a specified provider on Autonomous Database and applies to the ADMIN user or to the specified user.

Syntax

DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH(
    provider    IN VARCHAR2,
    username    IN VARCHAR2 DEFAULT 'ADMIN' );

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>provider</td>
<td>Specifies the type of provider. Valid values are: OCI, AWS, or AZURE.</td>
</tr>
<tr>
<td>username</td>
<td>Specifier the user to disable principal based authentication for.</td>
</tr>
<tr>
<td></td>
<td>A null value is valid for the username. If username is not specified, the</td>
</tr>
<tr>
<td></td>
<td>procedure applies for the &quot;ADMIN&quot; user.</td>
</tr>
</tbody>
</table>

Usage Notes

- When the provider value is AZURE and the username is ADMIN, the procedure disables Azure service principal based authentication on Autonomous Database and deletes the Azure application on the Autonomous Database instance.
- When the provider value is AZURE and the username is a user other than the ADMIN user, the procedure revokes the privileges from the specified user. The ADMIN user and other users that are enabled to use the Azure service principal can continue to use ADMIN.AZURE$PA and the application that is created for the Autonomous Database instance remains on the instance.
DISABLE_RESOURCE_PRINCIPAL Procedure

Disable resource principal credential for the database or for the specified schema.

Syntax

```
DBMS_CLOUD_ADMIN.DISABLE_RESOURCE_PRINCIPAL(
    username         IN VARCHAR2);
```

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Specifies an optional user name. The name of the database schema to remove resource principal access. If you do not supply a username, the username is set to ADMIN and the command removes the OCI$RESOURCE_PRINCIPAL credential.</td>
</tr>
</tbody>
</table>

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resource principal is already disabled</td>
<td>ORA-20031</td>
<td>If you attempt to disable the resource principal when it is already disabled.</td>
</tr>
</tbody>
</table>

Usage Notes

- Resource principal is not available with refreshable clones.
- You must set up a dynamic group and policies for the dynamic group before you call `DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL`.

  See the following for more information on creating policies, creating a dynamic group, and creating rules:
  - Use Resource Principal to Access Oracle Cloud Infrastructure Resources
  - Managing Dynamic Groups
  - Managing Policies

Example

```
EXEC DBMS_CLOUD_ADMIN.DISABLE_RESOURCE_PRINCIPAL();
```
PL/SQL procedure successfully completed.

SQL> select owner, credential_name from dba_credentials where credential_name = 'OCI$RESOURCE_PRINCIPAL';
No rows selected.

DROP_DATABASE_LINK Procedure

This procedure drops a database link.

Syntax

DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK(
   db_link_name      IN VARCHAR2,
   public_link       IN BOOLEAN DEFAULT);

Parameters

<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>
<tr>
<td>public_link</td>
<td>To run DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK with public_link set to TRUE, you must have the DROP PUBLIC DATABASE LINK system privilege. The default value for this parameter is FALSE.</td>
</tr>
</tbody>
</table>

Example

BEGIN
   DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK(
      db_link_name => 'SALES LINK');
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.</td>
</tr>
</tbody>
</table>

To find service names:

- Look in the tnsnames.ora file in the wallet.zip that you download from an Autonomous Database for your connection.
- Click DB Connection on the Oracle Cloud Infrastructure Console. In the Connection Strings area, each connection string includes a service_name entry that contains the connection string for the corresponding service. When both Mutual TLS (mTLS) and TLS connections are allowed, under TLS Authentication select TLS to view the TNS names and connection strings for connections with TLS authentication. See View TNS Names and Connection Strings for an Autonomous Database Instance for more information.
- Query V$SERVICES view. For example:

  SELECT name FROM V$SERVICES;

Usage Notes

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

Example

BEGIN
    DBMS_CLOUD_ADMIN.ENABLE_APP_CONT(
        service_name => 'nvthp2ht_adb1_high.adb.oraclecloud.com'
    );
END;
/

Verify the value as follows:

SELECT name, failover_type FROM DBA_SERVICES;

<table>
<thead>
<tr>
<th>NAME</th>
<th>FAILOVER_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvthp2ht_adb1_high.adb.oraclecloud.com</td>
<td>TRANSACTION</td>
</tr>
</tbody>
</table>

ENABLE_AWS_ARN Procedure

This procedure enables an Autonomous Database instance to use Amazon Resource Names (ARNs) to access AWS resources.
Syntax

DBMS_CLOUD_ADMIN.ENABLE_AWS_ARN(
    username     IN VARCHAR2 DEFAULT NULL,
    grant_option IN BOOLEAN DEFAULT FALSE);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Name of the user to enable to use Amazon Resource Names (ARNs). A null value is valid for the username. If username is not specified, the procedure applies for the &quot;ADMIN&quot; user.</td>
</tr>
<tr>
<td>grant_option</td>
<td>When username is supplied, if grant_option is TRUE the specified username can enable Amazon Resource Names (ARNs) usage for other users.</td>
</tr>
</tbody>
</table>

Example

BEGIN
    DBMS_CLOUD_ADMIN.ENABLE_AWS_ARN(
        username => 'adb_user');
END;
/

Usage Note

- You must be the ADMIN user to run the procedure DBMS_CLOUD_ADMIN.ENABLE_AWS_ARN.

See Use Amazon Resource Names (ARNs) to Access AWS Resources for more information.

ENABLE_EXTERNAL_AUTHENTICATION Procedure

Enable users to login to the database with external authentication schemes.

Syntax

DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type         IN VARCHAR2,
    force        IN BOOLEAN DEFAULT FALSE,
    params       IN CLOB DEFAULT NULL
);
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>type</strong></td>
<td>Specifies the external authentication type. Valid values: or .</td>
</tr>
<tr>
<td></td>
<td>• 'OCI_IAM'</td>
</tr>
<tr>
<td></td>
<td>• 'AZURE_AD'</td>
</tr>
<tr>
<td></td>
<td>• 'CMU'</td>
</tr>
<tr>
<td></td>
<td>• 'KERBEROS'</td>
</tr>
<tr>
<td><strong>force</strong></td>
<td>(Optional) Override a currently enabled external authentication scheme. Valid values are TRUE or FALSE.</td>
</tr>
<tr>
<td></td>
<td>The default value is FALSE.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>params</strong></td>
<td>A JSON string that provides additional parameters for the external authentication.</td>
</tr>
<tr>
<td><strong>CMU parameters:</strong></td>
<td></td>
</tr>
<tr>
<td>• location_uri:</td>
<td>specifies the cloud storage URI for the bucket where files required for CMU are stored. If you specify location_uri there is a fixed name directory object CMU_WALLET_DIR created in the database at the path 'cmu_wallet' to save the CMU configuration files. In this case, you do not need to supply the directory_name parameter.</td>
</tr>
<tr>
<td>• credential_name:</td>
<td>specifies the credentials that are used to download the CMU configuration files from the Object Store to the directory object. Default value is NULL which allows you to provide a Public, Preauthenticated, or pre-signed URL for Object Store bucket or subfolder.</td>
</tr>
<tr>
<td>• directory_name:</td>
<td>specifies the directory name where configuration files required for CMU are stored. If directory_name is supplied, you are expected to copy the CMU configuration files dsi.ora and cwallet.sso to this directory object.</td>
</tr>
<tr>
<td><strong>KERBEROS parameters:</strong></td>
<td></td>
</tr>
<tr>
<td>• location_uri:</td>
<td>specifies the cloud storage URI for the the bucket where files required for Kerberos are stored. If you specify location_uri there is a fixed name directory object KERBEROS_DIR created in the database to save the Kerberos configuration files. In this case, you do not need to supply the directory_name parameter.</td>
</tr>
<tr>
<td>• credential_name:</td>
<td>specifies the credential that are used to download Kerberos configuration files from the Object Store location to the directory object. Default value is NULL which allows you to provide a Public, Preauthenticated, or pre-signed URL for Object Store bucket or subfolder.</td>
</tr>
<tr>
<td>• directory_name:</td>
<td>specifies the directory name where files required for Kerberos are stored. If directory_name is supplied, you are expected to copy the Kerberos configuration files to this directory object.</td>
</tr>
<tr>
<td><strong>AZURE_AD parameters:</strong></td>
<td></td>
</tr>
<tr>
<td>• tenant_id:</td>
<td>Tenant ID of the Azure Account. Tenant Id specifies the Autonomous Database instance’s Azure AD application registration.</td>
</tr>
<tr>
<td>• application_id:</td>
<td>Azure Application ID created in Azure AD to assign roles/schema mappings for external authentication in the Autonomous Database instance.</td>
</tr>
<tr>
<td>• application_id_uri:</td>
<td>Unique URI assigned to the Azure Application. This is the identifier for the Autonomous Database instance. The name must be domain qualified (this supports cross tenancy resource access). The maximum length for this parameter is 256 characters.</td>
</tr>
</tbody>
</table>
Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid_ext_auth</td>
<td>ORA-20004</td>
<td>See the accompanying message for a detailed explanation.</td>
</tr>
</tbody>
</table>

Usage Notes

- With `type OCI_IAM`, if the resource principal is not enabled on the Autonomous Database instance, this routine enables resource principal with `DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL`.

- This procedure sets the system parameters `IDENTITY_PROVIDER_TYPE` and `IDENTITY_PROVIDER_CONFIG` to required users to access the instance with Oracle Cloud Infrastructure Identity and Access Management authentication and authorization.

Examples

Enable **OCI_IAM** Authentication

```plsql
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type => 'OCI_IAM',
    force => TRUE);
END;
/
```

PL/SQL procedure successfully completed.

Enable **CMU** Authentication for Microsoft Active Directory

You pass in a directory name that contains the CMU configuration files through `params JSON` argument.

```plsql
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type => 'CMU',
    force => TRUE,
    params => JSON_OBJECT('directory_name' value 'CMU_DIR'); // CMU_DIR directory object already exists
END;
/
```

PL/SQL procedure successfully completed.

You pass in a location URI pointing to an Object Storage location that contains CMU configuration files through `params JSON` argument.

```plsql
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type => 'CMU',
    params => JSON_OBJECT('location_uri' value 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
```
Enable Azure AD Authentication

BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
  type => 'AZURE_AD',
  force => TRUE,
  params => JSON_OBJECT('tenant_id' VALUE '....',
                        'application_id' VALUE '....',
                        'application_id_uri' VALUE '.....'));
END;
/

PL/SQL procedure successfully completed.

Enable Kerberos Authentication

You pass in a directory name that contains Kerberos configuration files through params JSON argument.

BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
  type => 'KERBEROS',
  force => TRUE,
  params => JSON_OBJECT('directory_name' value 'KERBEROS_DIR'); // KERBEROS_DIR directory object already exists
END;
/

PL/SQL procedure successfully completed.

You pass in a location URI pointing to an Object Storage location that contains Kerberos configuration files through params JSON argument.

BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
  type => 'KERBEROS',
  force => TRUE,
  params => JSON_OBJECT('location_uri' value 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o',
                        'credential_name' value 'my_credential_name');
END;
/

PL/SQL procedure successfully completed.
**ENABLE_FEATURE Procedure**

This procedure enables the specified feature on the Autonomous Database instance.

**Syntax**

```sql
DBMS_CLOUD_ADMIN.ENABLE_FEATURE(
    feature_name     IN VARCHAR2);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>feature_name</td>
<td>Name of the feature to enable. The only supported value is: 'JAVAVM'.</td>
</tr>
</tbody>
</table>

**Example**

```sql
BEGIN
    DBMS_CLOUD_ADMIN.ENABLE_FEATURE (
        feature_name => 'JAVAVM' );
END;
/```

**Usage Note**

- After you run `DBMS_CLOUD_ADMIN.ENABLE_FEATURE` with `feature_name` value 'JAVAVM', you must restart the Autonomous Database instance to install Oracle Java.

**ENABLE_PRINCIPAL_AUTH Procedure**

This procedure enables principal authentication on Autonomous Database for the specified provider and applies to the ADMIN user or the specified user.

**Syntax**

```sql
DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
    provider    IN VARCHAR2,
    username    IN VARCHAR2 DEFAULT 'ADMIN',
    params      IN CLOB DEFAULT NULL);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>provider</td>
<td>Specifies the type of provider. Valid values: OCI, AWS, or AZURE.</td>
</tr>
<tr>
<td>username</td>
<td>Name of the user who has principal authentication usage enabled. A null value is valid for the <code>username</code>. If <code>username</code> is not specified, the procedure applies for the &quot;ADMIN&quot; user.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>params</td>
<td>Specifies the configuration parameters. When the provider parameter is AWS or OCI, params is not required. The default value is NULL. When the provider parameter is AZURE, the following options are valid: azure_tenantid: with the value of the Azure tenant ID. grant_option: with the value specified as a boolean value: TRUE or FALSE. The default is FALSE. When TRUE and a username is specified, the specified user can enable Azure principal authentication for other users.</td>
</tr>
</tbody>
</table>

Usage Note

When the provider parameter is AZURE, the params parameter must include the azure_tenantid in the following cases:

- When DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH is called for the first time.
- When DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH is called for the first time after DBMS_CLOUD_ADMIN.DISABLE_PRINCIPAL_AUTH is called with the provider parameter AZURE and the username ADMIN.

Example

```sql
BEGIN
    DBMS_CLOUD_ADMIN.ENABLE_PRINCIPAL_AUTH(
        provider => 'AZURE',
        username => 'SCOTT',
        params   => JSON_OBJECT('azure_tenantid' value 'azure_tenantid'));
END;
/
```

ENABLE_RESOURCE_PRINCIPAL Procedure

Enable resource principal credential for the database or for the specified schema. This procedure creates the credential OCI$RESOURCE_PRINCIPAL.

Syntax

```sql
DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL(
    username IN VARCHAR2,
    grant_option IN BOOLEAN DEFAULT FALSE);
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Specifies an optional user name. The name of the database schema to be granted resource principal access. If you do not supply a username, the username is set to ADMIN.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>grant_option</td>
<td>When username is supplied, if grant_option is TRUE the specified username can enable resource principal usage for other users.</td>
</tr>
</tbody>
</table>

Exceptions

<table>
<thead>
<tr>
<th>Exception</th>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resource principal is already enabled</td>
<td>ORA-20031</td>
<td>If you attempt to enable the resource principal when it is already enabled.</td>
</tr>
</tbody>
</table>

Usage Notes

- You must call `DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL` with the `ADMIN` username or with no arguments before you call `DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL` with a username for a database user schema.
- Resource principal is not available with refreshable clones.
- You must set up a dynamic group and policies for the dynamic group before you call `DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL`.

  See the following for more information: for more information on creating a dynamic group and creating rules.

  See for more information on policies.
  - Use Resource Principal to Access Oracle Cloud Infrastructure Resources
  - Managing Dynamic Groups
  - Managing Policies

Example

```
EXEC DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL();

PL/SQL procedure successfully completed.

SQL> select owner, credential_name from dba_credentials where credential_name = 'OCI$RESOURCE_PRINCIPAL';

OWNER CREDENTIAL_NAME
------- ---------------
ADMIN OCI$RESOURCE_PRINCIPAL
```

REPLAY_WORKLOAD Procedure

This procedure enables the ADMIN user to initiate a workload replay. This procedure allows you to replay the capture files from an on-premises database or other cloud service databases.
Syntax

```sql
DBMS_CLOUD_ADMIN.REPLAY_WORKLOAD(
    location_uri IN VARCHAR2,
    credential_name IN VARCHAR2 DEFAULT NULL,
    synchronization IN BOOLEAN  DEFAULT TRUE,
    process_capture IN BOOLEAN  DEFAULT TRUE);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>location_uri</td>
<td>Specifies URI, that points to an Object Storage location that contains the</td>
</tr>
<tr>
<td></td>
<td>captured files. This parameter is mandatory.</td>
</tr>
<tr>
<td>credential_name</td>
<td>Specifies the credential to access the object storage bucket. If you do not</td>
</tr>
<tr>
<td></td>
<td>supply a credential_name value, the credential_name is set to NULL.</td>
</tr>
<tr>
<td>synchronization</td>
<td>Specifies the synchronization method used during workload replay.</td>
</tr>
<tr>
<td></td>
<td>• TRUE specifies that the synchronization is based on SCN.</td>
</tr>
<tr>
<td></td>
<td>• FALSE specifies that the synchronization is based on TIME.</td>
</tr>
<tr>
<td></td>
<td>If you do not supply a synchronization value, the synchronization is set to</td>
</tr>
<tr>
<td></td>
<td>TRUE.</td>
</tr>
<tr>
<td>process_capture</td>
<td>Specifies whether you need to specify process_capture value or not. It can</td>
</tr>
<tr>
<td></td>
<td>be set to FALSE only when you replay the same workload on the target</td>
</tr>
<tr>
<td></td>
<td>database repeatedly. If you do not supply a process_capture value, the</td>
</tr>
<tr>
<td></td>
<td>process_capture is set to TRUE.</td>
</tr>
</tbody>
</table>

Example to replay the workload from an on-premises or other cloud service database into an Autonomous Database:

```sql
BEGIN
    DBMS_CLOUD_ADMIN.REPLAY_WORKLOAD(
        location_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
        credential_name => 'CRED_TEST',
        synchronization => TRUE,
        process_capture => TRUE);
END;
/
```

When you run this procedure, it:

- Downloads the capture files from the Object Storage location specified in `location_uri`, processes the capture files based on the `process_capture` parameter value.
- Replays the captured workload based on the `synchronization` parameter value.
In this example, namespace-string is the Oracle Cloud Infrastructure object storage namespace and bucketname is the bucket name. See Understanding Object Storage Namespaces for more information.

See Navigate to Oracle Cloud Infrastructure Object Storage and Create Bucket for more information on Object Storage.

See Upload Files to Your Oracle Cloud Infrastructure Object Store Bucket for more information on uploading files to Object Storage.

The credential_name you use in this step is the credentials for the Object Store.

You don't need to create a credential to access Oracle Cloud Infrastructure Object Store if you enable resource principal credentials. See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

Usage Notes

• Run this procedure as the ADMIN user.

• Before you start replay, you should upload the cap and capfiles subdirectories containing capture files that are created during capture to the Object Storage location.

DBMS_CLOUD_ADMIN Exceptions

The following table describes exceptions for DBMS_CLOUD_ADMIN.

<table>
<thead>
<tr>
<th>Exception</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid_service</td>
<td>20001</td>
<td>An invalid service was specified.</td>
</tr>
<tr>
<td>service_not_exist</td>
<td>20002</td>
<td>A service specified does not exist.</td>
</tr>
<tr>
<td>default_service</td>
<td>20003</td>
<td>A service specified cannot be modified.</td>
</tr>
<tr>
<td>invalid_char_set</td>
<td>20029</td>
<td>Missing precondition or invalid (national) character set.</td>
</tr>
<tr>
<td>invalid_enc_key_attr</td>
<td>20030</td>
<td>Missing or invalid argument for key management.</td>
</tr>
<tr>
<td>Already Using Oracle Managed Key</td>
<td>000000</td>
<td>The Database is already using an Oracle managed key. You are trying to call the procedure while already using an Oracle managed key.</td>
</tr>
</tbody>
</table>

 Argument Provided for the procedure ORA-0000 An argument is provided for the procedure. Expected error message:No arguments required for this procedure.

DBMS_CLOUD_MACADM Package

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

```
EXEC DBMS_CLOUD_MACADM.DISABLE_DATABASE_VAULT;
```

DISABLE_USERMGMT_DATABASE_VAULT Procedure

This procedure disallows user management related operations for specified components on Autonomous Database with Oracle Database Vault enabled.

Syntax

```
DBMS_CLOUD_MACADM.DISABLE_USERMGMT_DATABASE_VAULT('component_name');
```

Where: `component_name` is the component name. Valid value is: APEX.

APEX is the Oracle APEX component.

Usage Notes

If you enable Oracle Database Vault with Autonomous Database and you want to enforce strict separation of duty to disallow user management related operations for the APEX, use the `DBMS_CLOUD_MACADM.DISABLE_USERMGMT_DATABASE_VAULT` procedure.

To use this procedure you must have the `DV_ACCTMGR` and `DV_ADMIN` roles.

Example

The following example disables user management for the APEX component:

```
EXEC DBMS_CLOUD_MACADM.DISABLE_USERMGMT_DATABASE_VAULT('APEX');
```

ENABLE_DATABASE_VAULT Procedure

This procedure enables Oracle Database Vault on Autonomous Database. To use this procedure you must have the `DV_OWNER` role.

Syntax

```
DBMS_CLOUD_MACADM.ENABLE_DATABASE_VAULT;
```

Usage Notes

After you run `DBMS_CLOUD_MACADM.ENABLE_DATABASE_VAULT` you must restart the Autonomous Database instance.

To use this procedure you must have the `DV_OWNER` role.
Example

The following example enables Oracle Database Vault:

BEGIN
  DBMS_CLOUD_MACADM.ENABLE_DATABASE_VAULT;
END;
/

ENABLE_USERMGMT_DATABASE_VAULT Procedure

This procedure allows user management with Oracle Database Vault enabled for specified components on Autonomous Database.

Syntax

DBMS_CLOUD_MACADM.ENABLE_USERMGMT_DATABASE_VAULT('component_name');

Where: component_name is the component name. Valid value is: APEX.
APEX is the Oracle APEX component.

Usage Notes

To use this procedure you must have the DV_ACCTMGR and DV_ADMIN roles.

Example

The following example enables user management for the APEX component:

EXEC DBMS_CLOUD_MACADM.ENABLE_USERMGMT_DATABASE_VAULT('APEX');

DBMS_CLOUD_REPO Package

The DBMS_CLOUD_REPO package provides for use of and management of cloud hosted code repositories from Oracle Database. Supported cloud code repositories include GitHub, AWS CodeCommit, and Azure Repos.

Topics

- DBMS_CLOUD_REPO Overview
- DBMS_CLOUD_REPO Data Structures
- DBMS_CLOUD_REPO Subprogram Groups
- Summary of DBMS_CLOUD_REPO Subprograms
DBMS_CLOUD_REPO Overview

The DBMS_CLOUD_REPO package provides easy access to files in Cloud Code (Git) Repositories, including: GitHub, AWS CodeCommit, and Azure Repos.

This package is a single interface for access to Multicloud Code repositories and allows you to upload SQL files to Git repositories or install SQL scripts directly from Cloud Code Repositories. This package also allows you to use a Cloud Code Repository to manage code versions for SQL scripts and to install or patch application code from Git repositories.

Concepts

- **Git Version Control System**: Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows.
- **Git Repository**: A Git repository is a virtual storage of your project. It allows you to save versions of your code, which you can access when needed.

Architecture

DBMS_CLOUD_REPO package provides four feature areas:

- **Repository Initialization with Generic Cloud Code Repository Handle**
  - Initialize a GitHub Code Repository
  - Initialize an AWS CodeCommit Code Repository
  - Initialize an Azure Repos Code Repository
- **Repository Management Operations**
  - Create a repository
  - Update a repository
  - List repositories
  - Delete a repository
- **Repository File Management Operations**
  - Upload a file to Code Repository from Oracle Database.
  - Download a file from Code Repository to Oracle Database.
  - Delete files from Code Repository.
  - List files from Code Repository.
- **SQL Install Operations**
  - Export Database object metadata DDL to repository.
  - Install SQL statements from a file in the Code Repository in Oracle Database.
  - Install SQL statements from a buffer.
DBMS_CLOUD_REPO Data Structures

The DBMS_CLOUD_REPO package defines record types and a generic JSON object type repo.

REPO JSON Object

A DBMS_CLOUD_REPO REPO is an opaque JSON object to represent a Cloud Code Repository of a specific cloud provider. A REPO object can be passed to different DBMS_CLOUD_REPO APIs. This opaque object ensures that DBMS_CLOUD_REPO procedures and functions are multcloud compatible; you do not have to change any code when you migrate from one Cloud Code Repository provider to another Cloud Code Repository.

DBMS_CLOUD_REPO Subprogram Groups

The DBMS_CLOUD_REPO package subprograms can be grouped into four categories: Initialization Operations, Repository Management Operations, File Operations, and SQL Install Operations.

Topics

• DBMS_CLOUD_REPO Initialization Operations
• DBMS_CLOUD_REPO Repository Management Operations
• DBMS_CLOUD_REPO File Operations
• DBMS_CLOUD_REPO SQL Install Operations

DBMS_CLOUD_REPO Initialization Operations

Lists the subprograms for initialization operations within the DBMS_CLOUD_REPO package.

<table>
<thead>
<tr>
<th>Subprogram</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT_AWS_REPO Function</td>
<td>This function initializes an AWS repository handle and returns an opaque type.</td>
</tr>
<tr>
<td>INIT_AZURE_REPO Function</td>
<td>This function initializes an Azure repository handle and returns an opaque type.</td>
</tr>
<tr>
<td>INIT_GITHUB_REPO Function</td>
<td>This function initializes a GitHub repository handle and returns an opaque type.</td>
</tr>
<tr>
<td>INIT_REPO Function</td>
<td>This function initializes a Cloud Code Repository handle and returns an opaque JSON object.</td>
</tr>
</tbody>
</table>

DBMS_CLOUD_REPO Repository Management Operations

Shows the subprograms for repository management operations within the DBMS_CLOUD_REPO package.

<table>
<thead>
<tr>
<th>Subprogram</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE_REPOSITORY Procedure</td>
<td>This procedure creates a Cloud Code Repository identified by the repo handle argument.</td>
</tr>
</tbody>
</table>
### DELETE_REPOSITORY Procedure

This procedure deletes the Cloud Code Repository identified by the `repo` handle argument.

### LIST_REPOSITORIES Function

This function lists all the Cloud Code Repositories identified by the `repo` handle argument.

### UPDATE_REPOSITORY Procedure

This procedure updates a Cloud Code repository identified by the `repo` handle argument. The procedure supports updating the name, description, or private visibility status, as supported by the Cloud Code repository.

### DBMS_CLOUD_REPO File Operations

#### DELETE_FILE Procedure

This procedure deletes a file from the Cloud Code repository identified by the `repo` handle argument.

#### GET_FILE Procedure and Function

The function downloads the contents of a file from the Cloud Code repository. The procedure allows you to download the contents of a file from the Cloud Code repository and save the file in a directory.

#### LIST_FILES Function

This function downloads a file from Cloud Code repository. Optionally, file content can be accessed from either a specific branch, tag or commit name. By default, the file is accessed from the default repository branch.

#### PUT_FILE Procedure

This procedure uploads a file to the Cloud Code repository identified by the `repo` handle argument. The procedure is overloaded to support either uploading a file from a directory object or uploading the contents from a CLOB to the repository file.

### DBMS_CLOUD_REPO SQL Install Operations

#### EXPORT_OBJECT Procedure

This procedure uploads the DDL metadata of a database object to the Cloud Code repository identified by the `repo` handle argument.

#### INSTALL_FILE Procedure

This procedure installs SQL statements from a file in the Cloud Code repository identified by the `repo` handle argument.

#### INSTALL_SQL Procedure

This procedure installs SQL statements from a buffer given as input.

### Summary of DBMS_CLOUD_REPO Subprograms

This section covers the DBMS_CLOUD_REPO subprograms provided with Autonomous Database.

The DBMS_CLOUD_REPO package is made up of the following:

- DBMS_CLOUD_REPO Initialization Operations
- DBMS_CLOUD_REPO Repository Management Operations
- DBMS_CLOUD_REPO File Operations
- DBMS_CLOUD_REPO SQL Install Operations
CREATE_REPOSITORY Procedure

This procedure creates a Cloud Code Repository identified by the `repo` handle argument.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.CREATE_REPOSITORY(
    repo                 IN   CLOB,
    description          IN   CLOB     DEFAULT NULL,
    private              IN   BOOLEAN  DEFAULT TRUE
);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repo</td>
<td>Specifies the repository handle. This parameter is supported for all cloud providers.</td>
</tr>
<tr>
<td>description</td>
<td>A short text description for the repository. This parameter is supported for GITHUB and AWS cloud provider.</td>
</tr>
<tr>
<td>private</td>
<td>Repository is private and only accessible with valid credentials. This parameter is only supported for the GITHUB cloud provider.</td>
</tr>
</tbody>
</table>

Example

```
BEGIN
    DBMS_CLOUD_REPO.CREATE_REPOSITORY(
        repo         => l_repo,
        description  => 'My test repo',
        private      => TRUE
    );
END;
/
```

DELETE_FILE Procedure

This procedure deletes a file from the Cloud Code repository identified by the `repo` handle argument.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.DELETE_FILE(
    repo              IN  CLOB,
    file_path         IN  VARCHAR2,
    branch_name       IN  VARCHAR2  DEFAULT NULL,
    commit_details    IN  CLOB      DEFAULT NULL
);
```
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repo</td>
<td>Specifies the repository handle.</td>
</tr>
<tr>
<td>file_path</td>
<td>File path to delete file in the repository.</td>
</tr>
<tr>
<td>branch_name</td>
<td>Delete file from a specific branch.</td>
</tr>
<tr>
<td>commit_details</td>
<td>Commit Details as a JSON document</td>
</tr>
</tbody>
</table>

    
    
    

Example

BEGIN
    DBMS_CLOUD_REPO.DELETE_FILE(
        repo => l_repo,
        file_path => 'scripts/test3.sql',
        branch_name => 'test_branch'
    );
END;
/

DELETE_REPOSITORY Procedure

This procedure deletes the Cloud Code Repository identified by the repo handle argument.

Syntax

PROCEDURE DBMS_CLOUD_REPO.DELETE_REPOSITORY(
    repo IN CLOB
);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repo</td>
<td>Specifies the repository handle.</td>
</tr>
</tbody>
</table>

Example

BEGIN
    DBMS_CLOUD_REPO.DELETE_REPOSITORY(
        repo => l_repo
    );
END;
/
**EXPORT_OBJECT Procedure**

This procedure uploads the DDL metadata of a database object to the Cloud Code repository identified by the repo handle argument. This procedure is an easy way to upload the metadata definition of a database object in single step.

**Syntax**

```sql
PROCEDURE DBMS_CLOUD_REPO.EXPORT_OBJECT(
    repo              IN  CLOB,
    file_path         IN  VARCHAR2,
    object_type       IN  VARCHAR2,
    object_name       IN  VARCHAR2 DEFAULT NULL,
    object_schema     IN  VARCHAR2 DEFAULT NULL,
    branch_name       IN  VARCHAR2 DEFAULT NULL,
    commit_details    IN  CLOB     DEFAULT NULL,
    append            IN  BOOLEAN  DEFAULT FALSE
);
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repo</td>
<td>Specifies the repository handle.</td>
</tr>
<tr>
<td>file_path</td>
<td>File path to upload object metadata in the repository.</td>
</tr>
<tr>
<td>object_type</td>
<td>Object type supported by DBMS_METADATA. See DBMS_METADATA: Object Types table for details.</td>
</tr>
<tr>
<td>object_name</td>
<td>Name of the database object to retrieve metadata.</td>
</tr>
<tr>
<td>object_schema</td>
<td>Owning schema of the database object.</td>
</tr>
<tr>
<td>branch_name</td>
<td>Put file to a specific branch.</td>
</tr>
<tr>
<td>commit_details</td>
<td>Commit Details as a JSON document:{&quot;message&quot;: &quot;Commit message&quot;, &quot;author&quot;: {&quot;name&quot;: &quot;Committing user name&quot;, &quot;email&quot;: &quot;Email of committing user&quot; } }</td>
</tr>
<tr>
<td>append</td>
<td>Append metadata DDL to existing file.</td>
</tr>
</tbody>
</table>

**Usage Note**

For customized control on the object DDL, you can use DBMS_METADATA.GET_DDL along with DBMS_CLOUD_REPO.PUT_FILE. In order to get metadata definition of the object, the current user must be privileged to retrieve the object metadata. See DBMS_METADATA for the security requirements of the package.

**Example**

```sql
BEGIN
    DBMS_CLOUD_REPO.EXPORT_OBJECT(
        repo => l_repo,
        object_type => 'PACKAGE',
        object_name => 'MYPACK',
        file_path => 'mypack.sql'
    );
```
GET_FILE Procedure and Function

The function downloads the contents of a file from the Cloud Code repository. The procedure allows you to download the contents of a file from the Cloud Code repository and save the file in a directory.

Syntax

FUNCTION DBMS_CLOUD_REPO.GET_FILE(
    repo              IN  CLOB,
    file_path         IN  VARCHAR2,
    branch_name       IN  VARCHAR2  DEFAULT NULL,
    tag_name          IN  VARCHAR2  DEFAULT NULL,
    commit_name       IN  VARCHAR2  DEFAULT NULL
) RETURN CLOB;

PROCEDURE DBMS_CLOUD_REPO.GET_FILE(
    repo              IN  CLOB,
    file_path         IN  VARCHAR2,
    directory_name    IN  VARCHAR2,
    target_file_name  IN  VARCHAR2  DEFAULT NULL,
    branch_name       IN  VARCHAR2  DEFAULT NULL,
    tag_name          IN  VARCHAR2  DEFAULT NULL,
    commit_name       IN  VARCHAR2  DEFAULT NULL
);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repo</td>
<td>Specifies the repository handle.</td>
</tr>
<tr>
<td>file_path</td>
<td>File path in the repository.</td>
</tr>
<tr>
<td>directory_name</td>
<td>Directory object name to save the file contents.</td>
</tr>
<tr>
<td>target_file_name</td>
<td>Target file name to save contents in directory.</td>
</tr>
<tr>
<td>branch_name</td>
<td>Get file from a specific branch.</td>
</tr>
<tr>
<td>tag_name</td>
<td>Get file from a specific Tag.</td>
</tr>
<tr>
<td>commit_name</td>
<td>Get file from a specific commit.</td>
</tr>
</tbody>
</table>

Example

BEGIN
    DBMS_CLOUD_REPO.GET_FILE(
        repo      =>  l_repo,
        file_path =>  'test3.sql',
        directory_name   =>  'DATA_PUMP_DIR',
        target_file_name =>  'test2.sql'
    );
END;
/*
INIT_AWS_REPO Function

This function initializes an AWS repository handle and returns an opaque type.

Syntax

```sql
FUNCTION DBMS_CLOUD_REPO.INIT_AWS_REPO(
    credential_name IN VARCHAR2,
    repo_name       IN VARCHAR2,
    region          IN VARCHAR2
) RETURN repo;
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>Credential object specifying AWS CodeCommit accesskey/secretkey.</td>
</tr>
<tr>
<td>repo_name</td>
<td>Specifies the repository name.</td>
</tr>
<tr>
<td>region</td>
<td>Specifies the AWS region for the CodeCommit repository.</td>
</tr>
</tbody>
</table>

Example

```sql
BEGIN
    :repo := DBMS_CLOUD_REPO.INIT_AWS_REPO(
        credential_name => 'AWS_CRED',
        repo_name       => 'my_repo',
        region          => 'us-east-1'
    );
END;
/
```

INIT AZURE_REPO Function

This function initializes an Azure repository handle and returns an opaque type. This function is only supported for Azure cloud provider.

Syntax

```sql
FUNCTION DBMS_CLOUD_REPO.INIT_AZURE_REPO(
    credential_name IN VARCHAR2,
    repo_name       IN VARCHAR2,
    organization    IN VARCHAR2,
    project         IN VARCHAR2
) RETURN repo;
```
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>Credential object specifying Azure, with a Username and Personal Access Token (PAT).</td>
</tr>
<tr>
<td>repo_name</td>
<td>Specifies the repository name.</td>
</tr>
<tr>
<td>organization</td>
<td>Specifies the Azure DevOps Organization.</td>
</tr>
<tr>
<td>project</td>
<td>Azure Team Project name.</td>
</tr>
</tbody>
</table>

Example

```
BEGIN
  :repo := DBMS_CLOUD_REPO.INIT_AZURE_REPO(
    credential_name => 'AZURE_CRED',
    repo_name       => 'my_repo',
    organization    => 'myorg',
    project         => 'myproj',
  );
END;
/
```

INIT_GITHUB_REPO Function

This function initializes a GitHub repository handle and returns an opaque type.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.INIT_GITHUB_REPO(
  credential_name IN VARCHAR2 DEFAULT NULL,
  repo_name       IN VARCHAR2,
  owner           IN VARCHAR2)
RETURN repo;
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>Credential object specifying GitHub.</td>
</tr>
<tr>
<td>repo_name</td>
<td>User Email and Personal Access Token (PAT).</td>
</tr>
<tr>
<td>owner</td>
<td>Specifies the repository owner.</td>
</tr>
</tbody>
</table>

Example

```
BEGIN
  :repo := DBMS_CLOUD_REPO.INIT_GITHUB_REPO(
    credential_name => 'GITHUB_CRED',
    repo_name       => 'my_repo',
    owner           => 'foo',
  );
END;
```
INIT_REPO Function

This function initializes a Cloud Code Repository handle and returns an opaque JSON object. This function is a generic interface to accept a JSON document, and avoids having to change code, you only need to change a JSON document, when moving a code repository from one Cloud Code repository to another Cloud Code repository.

Syntax

```sql
FUNCTION DBMS_CLOUD_REPO.INIT_REPO(
  params IN CLOB)
RETURN CLOB;
```

Parameters

<table>
<thead>
<tr>
<th>JSON Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>provider</td>
<td>Cloud code repository provider from the following:</td>
</tr>
<tr>
<td></td>
<td>DBMS_CLOUD_REPO.GITHUB_REPO ('GITHUB')</td>
</tr>
<tr>
<td></td>
<td>DBMS_CLOUD_REPO.AWS_REPO ('AWS')</td>
</tr>
<tr>
<td></td>
<td>DBMS_CLOUD_REPO.AZURE_REPO ('AZURE')</td>
</tr>
<tr>
<td>repo_name</td>
<td>Specifies the repository name.</td>
</tr>
<tr>
<td>owner</td>
<td>GitHub Repository Owner. DBMS_CLOUD_REPO.PARAM_OWNER This parameter is only applicable for GitHub cloud provider.</td>
</tr>
<tr>
<td>region</td>
<td>AWS Repository Region DBMS_CLOUD_REPO_PARAM_REGION This parameter is only applicable for AWS cloud provider.</td>
</tr>
<tr>
<td>organization</td>
<td>Azure Organization DBMS_CLOUD_REPO_PARAM_ORGANIZATION This parameter is only applicable for Azure cloud provider.</td>
</tr>
<tr>
<td>project</td>
<td>Azure Team Project DBMS_CLOUD_REPO_PARAM_PROJECT This parameter is only applicable for Azure cloud provider</td>
</tr>
</tbody>
</table>

Example

```sql
BEGIN
  :repo := DBMS_CLOUD_REPO.INIT_REPO(  
    params => JSON_OBJECT('credential_name' value 'mycred',  
      'repo_name' value 'myrepo',  
      'repo_owner' value 'foo')
  );
END;
```

INSTALL_FILE Procedure

This procedure installs SQL statements from a file in the Cloud Code repository identified by the repo handle argument.
Syntax

PROCEDURE DBMS_CLOUD_REPO.INSTALL_FILE(
   repo              IN  CLOB,
   file_path         IN  VARCHAR2,
   branch_name       IN  VARCHAR2  DEFAULT NULL,
   tag_name          IN  VARCHAR2  DEFAULT NULL,
   commit_name       IN  VARCHAR2  DEFAULT NULL,
   stop_on_error     IN  BOOLEAN   DEFAULT TRUE
);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repo</td>
<td>Specifies the repository handle.</td>
</tr>
<tr>
<td>file_path</td>
<td>File path in the repository.</td>
</tr>
<tr>
<td>branch_name</td>
<td>Branch to install file from a specific branch.</td>
</tr>
<tr>
<td>tag_name</td>
<td>Tag to install file from a specific Tag.</td>
</tr>
<tr>
<td>commit_name</td>
<td>Commit ID to install file from a specific commit.</td>
</tr>
<tr>
<td>stop_on_error</td>
<td>Stop executing the SQL statements on first error.</td>
</tr>
</tbody>
</table>

Usage Notes

- The scripts are intended as schema install scripts and not as generic SQL scripts:
  - Scripts cannot contain SQL*Plus client specific commands.
  - Scripts cannot contain bind variables or parameterized scripts.
  - SQL statements must be terminated with a slash on a new line (/).
  - Scripts can contain DDL, DML PL/SQL statements, but direct SELECT statements are not supported. Using SELECT within a PL/SQL block is supported.

Any SQL statement that can be run using EXECUTE IMMEDIATE will work if it does not contain bind variables or defines.

Example

BEGIN
   DBMS_CLOUD_REPO.INSTALL_FILE(
      repo => l_repo,
      file_path => 'test3.sql',
      stop_on_error => FALSE
   );
END;
/

INSTALL_SQL Procedure

This procedure installs SQL statements from a buffer given as input.
Syntax

PROCEDURE DBMS_CLOUD_REPO.INSTALL_SQL(
    content       IN  CLOB,
    stop_on_error IN  BOOLEAN  DEFAULT TRUE
);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>content</td>
<td>Is the CLOB containing the SQL statements to run.</td>
</tr>
<tr>
<td>stop_on_error</td>
<td>Stop executing the SQL statements on first error.</td>
</tr>
</tbody>
</table>

Usage Notes

- The scripts are intended as schema install scripts and not as generic SQL scripts:
  - Scripts cannot contain SQL*Plus client specific commands.
  - Scripts cannot contain bind variables or parameterized scripts.
  - SQL statements must be terminated with a slash on a new line (/).
  - Scripts can contain DDL, DML PLSQL statements, but direct SELECT statements are not supported. Using SELECT within a PL/SQL block is supported.

Any SQL statement that can be run using EXECUTE IMMEDIATE will work if it does not contain bind variables or defines.

Example

BEGIN
    DBMS_CLOUD_REPO.INSTALL_SQL(
        content => 'create table t1 (x varchar2(30))' || CHR(10) || '/','
        stop_on_error => FALSE
    );
END;
/

LIST_FILES Function

This function downloads a file from Cloud Code repository. Optionally, file content can be accessed from either a specific branch, tag or commit name. By default, the file is accessed from the default repository branch. The results include the file names and additional metadata about the files.

Syntax

FUNCTION DBMS_CLOUD_REPO.LIST_FILES(
    repo   IN  CLOB,
    path   IN  VARCHAR2  DEFAULT NULL,
    branch_name IN  VARCHAR2  DEFAULT NULL,
);
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repo</td>
<td>Specifies the repository handle. This parameter is supported by all cloud providers.</td>
</tr>
<tr>
<td>path</td>
<td>List files under the specified subfolder path in the repository.</td>
</tr>
<tr>
<td>branch_name</td>
<td>List files from a specific branch.</td>
</tr>
<tr>
<td>tag_name</td>
<td>List files from a specific Tag.</td>
</tr>
<tr>
<td>commit_name</td>
<td>List files from a specific commit.</td>
</tr>
</tbody>
</table>

Usage Notes

- This is a pipelined table function with return type as list_file_ret_tab.
- DBMS_CLOUD_REPO.LIST_FILES returns the columns: id, name, url, and bytes.

Example

```sql
SELECT name FROM DBMS_CLOUD_REPO.LIST_FILES(repo => l_repo);
```

NAME
-------------------------
test3.sql

LIST_REPOSITORIES Function

This function lists all the Cloud Code Repositories identified by the repo handle argument. The results include the repository names and additional metadata about the repositories.

Syntax

```sql
FUNCTION DBMS_CLOUD_REPO.LIST_REPOSITORIES(
    repo        IN CLOB
) RETURN list_repo_ret_tab PIPELINED PARALLEL_ENABLE;
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repo</td>
<td>Specifies the repository handle. This parameter is supported by all cloud providers.</td>
</tr>
<tr>
<td>description</td>
<td>A short text description for the repository. This parameter is supported by the GITHUB and AWS cloud providers.</td>
</tr>
<tr>
<td>private</td>
<td>Repository is private and only accessible with valid credentials. This parameter is supported for the GITHUB cloud provider.</td>
</tr>
</tbody>
</table>
Usage Notes

- This is a pipelined table function with return type as `list_repo_ret_tab`.
- `DBMS_CLOUD_REPO.LIST_REPOSITORIES` returns the columns: id, name, owner, description, private, url, bytes, created, and last_modified.

Example

```sql
SELECT name, description FROM DBMS_CLOUD_REPO.LIST_REPOSITORIES(:repo);
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestRepo1</td>
<td>My test repo</td>
</tr>
</tbody>
</table>

PUT_FILE Procedure

This procedure uploads a file to the Cloud Code repository identified by the `repo` handle argument. The procedure is overloaded to support either uploading a file from a directory object or uploading the contents from a CLOB to the repository file.

Syntax

```sql
PROCEDURE DBMS_CLOUD_REPO.PUT_FILE(
    repo IN CLOB,
    file_path IN VARCHAR2,
    contents IN BLOB,
    branch_name IN VARCHAR2 DEFAULT NULL,
    commit_details IN CLOB DEFAULT NULL
);
```

```sql
PROCEDURE DBMS_CLOUD_REPO.PUT_FILE(
    repo IN CLOB,
    file_path IN VARCHAR2,
    directory_name IN VARCHAR2,
    source_file_name IN VARCHAR2 DEFAULT NULL,
    branch_name IN VARCHAR2 DEFAULT NULL,
    commit_details IN CLOB DEFAULT NULL
);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repo</td>
<td>Specifies the repository handle.</td>
</tr>
<tr>
<td>file_path</td>
<td>File path to upload file in the repository.</td>
</tr>
<tr>
<td>contents</td>
<td>BLOB containing the file contents.</td>
</tr>
<tr>
<td>directory_name</td>
<td>Directory object name containing the file name.</td>
</tr>
<tr>
<td>source_file_name</td>
<td>Source file name to upload to repository.</td>
</tr>
<tr>
<td>branch_name</td>
<td>Put file to a specific branch.</td>
</tr>
</tbody>
</table>
**Parameter** | **Description**
--- | ---
**commit_details** | Commit Details as a JSON document:
{
    "message": "Commit message", 
    "author": {
        "name": "Committing user name", 
        "email": "Email of committing user"
    }
}

**Example**

BEGIN
    DBMS_CLOUD_REPO.PUT_FILE(
        repo   => l_repo,
    );
END;
/

**UPDATE_REPOSITORY Procedure**

This procedure updates a Cloud Code repository identified by the `repo` handle argument. UPDATE_REPOSITORY supports updating the name, description, or private visibility status, as supported by the Cloud Code repository.

**Syntax**

PROCEDURE DBMS_CLOUD_REPO.UPDATE_REPOSITORY(
    repo                 IN OUT  CLOB,
    new_name             IN      VARCHAR2 DEFAULT NULL,
    description          IN      CLOB     DEFAULT NULL,
    private              IN      BOOLEAN  DEFAULT NULL
);

**Parameters**

**Parameter** | **Description**
--- | ---
repo | Specifies the repository handle. This parameter is supported for all cloud providers.
new_name | New name for repository. This parameter is supported for all cloud providers.
description | A short text description for the repository. This parameter is supported for GITHUB and AWS cloud providers.
private | Repository is private and only accessible with valid credentials. This parameter is supported for the GITHUB cloud provider.

**Example**

BEGIN
    DBMS_CLOUD_REPO.UPDATE_REPOSITORY(
        repo        => l_repo,
        new_name    => 'repo2'
    );
DBMS_DCAT Package

The DBMS_DCAT package provides functions and procedures to help Autonomous Database users leverage the data discovery and centralized metadata management system of OCI Data Catalog.

Data Catalog harvests metadata from a data lake’s object storage assets. The harvesting process creates logical entities, which can be thought of as tables with columns and associated data types. DBMS_DCAT procedures and functions connect Autonomous Database to Data Catalog and then synchronize the assets with the database, creating protected schemas and external tables. You can then query object store using those external tables, easily joining external data with data stored in Autonomous Database. This dramatically simplifies the management process; there is a single, centrally managed metadata store that is shared across multiple OCI services (including Autonomous Databases). There are also Autonomous Database dictionary views that allow you to inspect the contents of Data Catalog using SQL, and show you how these Data Catalog entities map to your Autonomous Database schemas and tables.

Topics
- Managing the Data Catalog Connection
- Running Synchronizations
- Data Catalog Views

Managing the Data Catalog Connection

Creating, querying and dropping a Data Catalog connection can be performed with managing the Data Catalog connection procedures.

Topics
- Data Catalog Users and Roles
- Required Credentials and IAM Policies
- SET_DATA_CATALOG_CREDENTIAL Procedure
- SET_OBJECT_STORE_CREDENTIAL Procedure
- SET_DATA_CATALOG_CONN Procedure
- UNSET_DATA_CATALOG_CONN Procedure

Data Catalog Users and Roles

Data Catalog Users
- Synced users/schemas
  The synced external tables are organized into database schemas corresponding to Data Asset/Bucket combinations, or according to custom properties set by the user. The synced schemas are automatically created/dropped during Data Catalog
synchronization. They are created as no authentication users without the CREATE SESSION privilege. The synced schemas are also created using the protected clause, so that they cannot be altered by local users (not even the PDB admin) and can only be modified through the synchronization.

- **User dcat_admin**
  User dcat_admin is a local database user that can run a sync and grant READ privilege on synced tables to other users or roles. The user is created as a no authentication user without the CREATE SESSION privilege.

- **Local users**
  Database users querying the external tables must be explicitly granted READ privileges on the synced external tables by users dcat_admin or ADMIN. By default, after the sync is completed, only users dcat_admin and ADMIN have access to the synced external tables.

**Data Catalog Roles**

- **dcat_sync**
  The dcat_sync role has all the required privileges for using the DBMS_DCAT package. Users must have this role to be able to use the API for navigating the Data Catalog and running the sync.

**Required Credentials and IAM Policies**

This topic describes the Oracle Cloud Infrastructure Identity and Access Management (IAM) user credentials and policies required to give Autonomous Database users permission to manage Data Catalog and to read from object storage.

The following are required:

- A credential object with permission to manage a Data Catalog instance is required. Credential objects containing OCI native authentication or resource principals credentials are supported. Credential objects based on authentication token user principals are not supported.
  For information on managing credentials, see DBMS_CLOUD for Access Management.

  For OCI native authentication examples, see Example: Creating an OCI Native Authentication Credential Object and Autonomous Database Now Supports Accessing the Object Storage with OCI Native Authentication.

  For examples using resource principal, see Example: Using Autonomous Database Resource Principal and Accessing Oracle Cloud Infrastructure Resources from Your Autonomous Database using Resource Principal.

- The manage Data Catalog privilege is required in order for Autonomous Database to add custom properties to the Data Catalog namespace. These privileges allow you to override schema names, table names, column names and more.
  For further information on Data Catalog permissions, see Permissions Required for Each API Operation.

- The read object storage privilege on buckets is required so that Autonomous Database can query data files.
  For further Oracle Object Storage Policy Examples, see Policy Examples.
Example: Creating an OCI Native Authentication Credential Object

In this example, we create an OCI native authentication credential that can be used when creating a data catalog or an object store credential object. For more details, see DBMS_DCAT SET_DATA_CATALOG_CREDENTIAL Procedure and DBMS_DCAT SET_OBJECT_STORE_CREDENTIAL Procedure respectively.

In OCI native authentication, the DBMS_CLOUD.CREATE_CREDENTIAL procedure includes these parameters: credential_name, user_ocid, tenancy_ocid, private_key, and fingerprint. See DBMS_CLOUD CREATE_CREDENTIAL Procedure for a complete description of this procedure.

The credential_name is the name of the credential object. The user_ocid and tenancy_ocid parameters correspond to the user's and tenancy's OCIDs respectively.

The private_key parameter specifies the generated private key in PEM format. Private keys created with a passphrase are not supported. Therefore, we need to make sure we generate a key with no passphrase. See How to Generate an API Signing Key for more details on how to create a private key with no passphrase. Also, the private key that we provide for this parameter must only contain the key itself without any header or footer (e.g. '-----BEGIN RSA PRIVATE KEY-----', '-----END RSA PRIVATE KEY-----').

The fingerprint parameter specifies the fingerprint that is obtained either after uploading the public key to the console, or using the OpenSSL commands. See How to Upload the Public Key and How to Get the Key's Fingerprint for further details on obtaining the fingerprint.

Once all the necessary information is gathered and the private key is generated, we're ready to run the following CREATE_CREDENTIAL procedure:

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL (
        credential_name => 'OCI_NATIVE_CRED',
        user_ocid => 'ocid1.user.oc1..aaaaaaaatfn77fe3fxux3o5lego7glqjejrzsqsrs64f4jsjrhsksqzndq',
        tenancy_ocid => 'ocid1.tenancy.oc1..aaaaaaaapwkfqz3upqklvmelbm3j77nn3y7uqmlsod75rea5zmtmb1574ve6a',
        private_key => 'MIIEogIBAAKCAQEA...t9SH72x7a5iV7QZJS5WeFLMUEv+YbYAjnXK+dOnPQtkhOb1QwCEY3Hsbl7Xz7o=',
END;
/
```

After creating the credential object, it displays in the dba_credentials table:

```
SELECT owner, credential_name
FROM dba_credentials
WHERE credential_name LIKE '%NATIVE%';

OWNER CREDENTIAL_NAME
```
ADMIN OCI_NATIVE_CRED

Example: Using Autonomous Database Resource Principal

In this example, a dynamic group is created that includes appropriate resource members, the
dynamic group is given permission to manage a Data Catalog, and then the dynamic group is
given permission to read from object storage.

1. Create a dynamic group named adb-grp-1. Add a matching rule to adb-grp-1 that
   includes the Autonomous Database instance with OCID
   ocid1.autonomousdatabase.oc1.iad.abuwcljr...fjkfe as a resource member.
   Dynamic group matching rule:
   
   resource.id = 'ocid1.autonomousdatabase.oc1.iad.abuwcljr...fjkfe'

2. Define a policy granting the adb-grp-1 dynamic group full access to the Data Catalog
   instances, in the mycompartment compartment.
   
   allow dynamic-group adb-grp-1 to manage data-catalog-family in
   compartment mycompartment

3. Define a policy that allows the adb-grp-1 dynamic group to read any bucket in the
   compartment named mycompartment.
   
   allow dynamic-group adb-grp-1 to read objects in compartment mycompartment

Example: Using User Principals

In this example, user1 is a member of the group adb-admins. All members of this group are
given permission to manage all data catalogs in mycompartment, and to read from object-
store in mycompartment.

1. Allow users that are members of adb-admins to manage all data catalogs within
   mycompartment.
   
   allow group adb-admins to manage data-catalog-family in compartment
   mycompartment

2. Allow users that are members of adb-admins to read any object in any bucket within
   mycompartment.
   
   allow group adb-admins to read objects in compartment mycompartment

SET_DATA_CATALOG_CREDENTIAL Procedure

This procedure sets the Data Catalog access credential used for all accesses to the Data
Catalog.

This credential must have Manage Data Catalog permissions; see Data Catalog Policies. The
default is the resource principal; see Access Cloud Resources by Configuring Policies and
Roles.
Syntax

PROCEDURE DBMS_DCAT.SET_DATA_CATALOG_CREDENTIAL(
    credential_name VARCHAR2(128) DEFAULT NULL);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>(Optional) The credential used for accessing the Data Catalog.</td>
</tr>
</tbody>
</table>

SET_OBJECT_STORE_CREDENTIAL Procedure

This procedure sets the credential that is used by the external tables for accessing the object store. Changing the Object Store access credential alters all existing synced tables to use the new credential.

Syntax

PROCEDURE DBMS_DCAT.SET_OBJECT_STORE_CREDENTIAL(
    credential_name VARCHAR2(128) DEFAULT NULL);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>(Optional) The credential used by the external tables for accessing the Object Store.</td>
</tr>
</tbody>
</table>

SET_DATA_CATALOG_CONN Procedure

This procedure creates a connection to a given Data Catalog. This is required to synchronize metadata with Data Catalog. An Autonomous Database instance can connect to a single Data Catalog instance. You only need to call this procedure once to set the connection. As part of the connection process, Autonomous Database adds custom properties to Data Catalog. These custom properties are accessible to Data Catalog users and allow you to override default names (for schemas, tables and columns) and column data types.

Syntax

PROCEDURE DBMS_DCAT.SET_DATA_CATALOG_CONN (
    region VARCHAR2(4000),
    endpoint VARCHAR2(4000) DEFAULT NULL,
    catalog_id VARCHAR2
);
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>region</td>
<td>The data catalog region.</td>
</tr>
<tr>
<td>endpoint</td>
<td>(Optional) The data catalog endpoint. Default is NULL.</td>
</tr>
<tr>
<td>catalog_id</td>
<td>The unique Oracle Cloud Identifier for the Data Catalog instance.</td>
</tr>
</tbody>
</table>

Example: Connecting with known OCID

In this example, Autonomous Database is connecting to Data Catalog in the `uk-london-1` region. The `catalog_id` parameter uses the Oracle Cloud Identifier (`ocid`) for the Data Catalog instance.

```
BEGIN
  DBMS_DCAT.SET_DATA_CATALOG_CONN(
    region=>'uk-london-1',
    catalog_id=>'ocid1.datacatalog.oc1.uk-london-1');
END;
/
```

UNSET_DATA_CATALOG_CONN Procedure

This procedure removes an existing data catalog connection.

**Note:**

Invoking this procedure drops all of the protected schemas and external tables that were created as part of previous synchronizations. It does not impact the metadata in Data Catalog.

**Syntax**

```
PROCEDURE DBMS_DCAT.UNSET_DATA_CATALOG_CONN;
```

Running Synchronizations

Running a synchronization, creating and dropping a synchronization job, and dropping synchronized schemas can be performed with these procedures.

**Note:**

On April 4, 2022, the `sync_option` and `grant_read` parameters were added to the `DBMS_DCAT.RUN_SYNC` procedure. To ensure correct performance of scheduled sync jobs created prior to that date, you need to drop and recreate the scheduled sync jobs. See `DBMS_DCAT.DROP_SYNC_JOB` Procedure and `DBMS_DCAT.CREATE_SYNC_JOB` Procedure.
Topics

- RUN_SYNC Procedure
- CREATE_SYNC_JOB Procedure
- DROP_SYNC_JOB Procedure
- DROP_SYNCED_SCHEMAS Procedure

RUN_SYNC Procedure

This procedure runs a synchronization operation and is the entry point to the synchronization. As an input, it takes lists of selected Data Catalog assets, folders and entities and materializes them by creating, dropping, and altering external tables.

The `sync_option` parameter specifies which operation the RUN_SYNC procedure performs: SYNC, DELETE or REPLACE. The operation is performed over entities within the scope of the `synced_objects` parameter.

Every call to the RUN_SYNC procedure returns a unique `operation_id` that can be used to query the USER_LOAD_OPERATIONS view to obtain information about the status of the sync and the corresponding `log_table`. The DBMS_DCAT$SYNC_LOG view can be queried for easy access to the `log_table` for the last sync operation executed by the current user. For further details, see DBMS_DCAT$SYNC_LOG View, and Monitoring and Troubleshooting Loads.

Note:

On April 4, 2022, the `sync_option` and `grant_read` parameters were added to the RUN_SYNC procedure. To ensure correct performance of scheduled sync jobs created prior to that date, you need to drop and recreate the scheduled sync jobs. See DBMS_DCAT.DROP_SYNC_JOB Procedure and DBMS_DCAT.CREATE_SYNC_JOB Procedure.

Synchronizing Partitioned Logical Entities

The RUN_SYNC procedure creates a partitioned external table for each logical entity when all three of the following apply:

1. The logical entity has one or more partitioned attributes.
2. The logical entity is derived from a prefix-based filename pattern. Partitioned logical entities derived from regex-based patterns are not supported.
3. The logical entity is based on partitioned data that follows the hive-style or non-hive folder format. Logical entities based on partitioned data that follow the non-hive style format using object names are not supported.
   - Example 1. Logical entities based on harvested objects that follow the Hive style partitioning format with prefix-based filename patterns.

   Consider the following objects:

   Bucket: MYBUCKET
   cluster1/db1.db/sales/country=USA/year=2020/month=01/sales1.csv
Harvesting the bucket using a filename pattern with a starting folder prefix of cluster1/db1.db generates a logical entity named SALES with three partition attributes: country, year, and month. The type for partitioned attributes is Partition while the type for non-partitioned attributes is Primitive.

- Example 2. Logical entities based on harvested objects that follow the non-Hive style partitioning format with prefix-based filename patterns.

  Consider the following objects:

  **Bucket: MYBUCKET**
  
  cluster2/db2.db/sales/USA/2020/01/sales1.csv  
  cluster2/db2.db/sales/USA/2020/01/sales2.csv  
  cluster2/db2.db/sales/USA/2020/02/sales1.csv

  Harvesting the bucket using a filename pattern with a starting folder prefix of cluster2/db2.db generates a logical entity named SALES with three partition attributes: name0, name1, and name2. The only difference between the generated logical entity compared to Example 1, is that the names of partitioned attributes are auto generated, while in Example 1 they are extracted from the URL (country, year, and month respectively).

  For a complete end to end example of synchronizing partitioned logical entities, see Example: A Partitioned Data Scenario.

### Incremental Synchronization of Partitioned Logical Entities

Every call to the RUN_SYNC procedure specifies a set of logical entities to be synced with the database. When a logical entity is present in two RUN_SYNC calls, the second call preserves and possibly alters existing external tables. The following table shows which logical entity changes are supported when the logical entity is partitioned:

<table>
<thead>
<tr>
<th>Logical Entity Change</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition, removal, or update of a partition</td>
<td>All partitions of the external partitioned table are updated, regardless of whether a change has been detected by the data catalog.</td>
</tr>
<tr>
<td>Addition of a partitioned attribute</td>
<td>Adding a partitioned column to an external partitioned table is not supported. An exception is raised.</td>
</tr>
<tr>
<td>Deletion of a partitioned attribute</td>
<td>Dropping a partitioned column from an external partitioned table is not supported. An exception is raised.</td>
</tr>
<tr>
<td>Renaming of a partitioned attribute</td>
<td>Renaming a partitioned column in an external partitioned table is not supported. An exception is raised.</td>
</tr>
</tbody>
</table>

### Syntax

```sql
PROCEDURE DBMS_DCAT.RUN_SYNC (  
    synced_objects   IN  CLOB,  
    sync_option      IN VARCHAR2 DEFAULT 'SYNC',  
    error_semantics  IN VARCHAR2 DEFAULT 'SKIP_ERRORS',  
    log_level        IN VARCHAR2 DEFAULT 'INFO',  
    grant_read       IN VARCHAR2 DEFAULT NULL,  
);```
PROCEDURE DBMS_DCAT.RUN_SYNC (  
synced_objects   IN  CLOB,  
sync_option      IN VARCHAR2 DEFAULT 'SYNC',  
error_semantics  IN VARCHAR2 DEFAULT 'SKIP_ERRORS',  
log_level        IN VARCHAR2 DEFAULT 'INFO',  
grant_read       IN VARCHAR2 DEFAULT NULL,  
operation_id     OUT NOCOPY NUMBER
);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>synced_objects</td>
<td>This parameter specifies a set of entities in multiple granularity: data assets, folders (Object Store buckets) or logical entities. It contains an asset_list that is either an array of asset objects or an array containing a single *** string that stands for 'sync all (object store) data assets in the catalog'.</td>
</tr>
</tbody>
</table>
| sync_option    | (Optional) There are three options:  
- **SYNC** (Default) - This option ensures that what is in the data catalog, over the synced_objects scope, is represented in the Autonomous Database. If a logical entity was deleted from the data catalog, since the last sync operation, then it is deleted in the Autonomous Database. The following operations are performed over the synced_objects scope:  
  - Adds tables for new data catalog entities  
  - Removes tables for deleted data catalog entities  
  - Updates properties (such as name, columns and data types) for existing tables  
- **DELETE** - Deletes tables within the synced_objects scope.  
- **REPLACE** - Replaces all currently synced objects with the objects within the synced_objects scope. |
| error_semantics | (Optional) This parameter specifies the error behavior. If set to SKIP_ERRORS, the sync attempts to continue despite errors encountered for individual entities. If set to STOP_ON_ERROR, the procedure fails on the first encountered error. The default is SKIP_ERRORS. |
| log_level      | (Optional) This parameter specifies the following values in increasing level of logging detail: (OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE, ALL). The default is INFO. |
| grant_read     | (Optional) This parameter is a list of users/roles that are automatically granted READ privileges on all external tables processed by this invocation of RUN_SYNC. All users/roles in the grant_read list are given READ privileges on all new or already existing external tables that correspond to entities specified by the synced_objects parameter. The RUN_SYNC procedure preserves already granted privileges on synced external tables. |
| operation_id   | (Optional) This parameter is used to find the corresponding entry in USER_LOAD_OPERATIONS for the sync and determine the name of the log table.  
Note: A version of RUN_SYNC that does not return an operation_id is available so users can query USER_LOAD_OPERATIONS for the latest sync. |
Example: Synchronize All Data Catalog Entities

In the following example, all Data Catalog entities are synchronized.

EXEC DBMS_DCAT.RUN_SYNC(synced_objects=>'"asset_list":['"*"']);

Example: \texttt{synced\_objects} Parameter for Synchronizing All Data Assets

The following is an example \texttt{synced\_objects} parameter for synchronizing all (Object Storage) data assets in the Data Catalog.

{"asset\_list": ["*"]}

Example: \texttt{synced\_objects} Parameter for Synchronizing Specific Data Assets

The following is an example \texttt{synced\_objects} parameter for synchronizing two data assets.

{"asset\_list": [
    {
        "asset\_id":"0b320de9-8411-4448-91fb-9e2e7f78fd5f"
    },
    {
        "asset\_id":"0b320de9-8411-4448-91fb-9e2e7f74523"
    }
]}

Example: \texttt{synced\_objects} Parameter for Synchronizing Specific Entities within a Data Asset

The following shows an example \texttt{synced\_objects} parameter for synchronizing two entities within the data asset.

{"asset\_list": [
    {
        "asset\_id":"0b320de9-8411-4448-91fb-9e2e7f78fd5f",
        "folder\_list": ["f1", "f2"
    }
]}

Example: \texttt{synced\_objects} Parameter for Synchronizing Specific Folders and Entities within a Data Asset

The following shows an example \texttt{synced\_objects} parameter for synchronizing two folders and two entities within the data asset.

{"asset\_list": [
    {
        "asset\_id":"0b320de9-8411-4448-91fb-9e2e7f78fd5f",
        "entity\_list": [
            "entity\_id": "0b320de9-8411-4448-91fb-9e2e7f78fd5f","entity\_id": "0b320de9-8411-4448-91fb-9e2e7f78fd5f"
        ]
    }
]}

CREATE_SYNC_JOB Procedure

This procedure creates a scheduler job to invoke `RUN_SYNC` periodically.

It takes as input the set of objects to be synced, the error semantics and the log level, as described in the `RUN_SYNC` API and a repeat interval.

There can only be a single sync job. The `CREATE_SYNC_JOB` procedure fails if another job is already specified, unless the force parameter is set to `TRUE`. If force is set to `TRUE` the previous job is dropped.

If a scheduler job attempts to run while another sync is in progress, the scheduler job fails.

**Note:**

On April 4, 2022, the `sync_option` and `grant_read` parameters were added to the `RUN_SYNC` procedure. To ensure correct performance of scheduled sync jobs created prior to that date, you need to drop and recreate the scheduled sync jobs. See DBMS_DCAT.DROP_SYNC_JOB Procedure and DBMS_DCAT.CREATE_SYNC_JOB Procedure.

### Syntax

```sql
PROCEDURE DBMS_DCAT.CREATE_SYNC_JOB (  
    synced_objects   IN CLOB,  
    error_semantics  IN VARCHAR2 DEFAULT 'SKIP_ERRORS',  
    log_level        IN VARCHAR2 DEFAULT 'INFO',  
    repeat_interval  IN VARCHAR2,  
    force            IN VARCHAR2 DEFAULT 'FALSE',  
    grant_read       IN VARCHAR2 DEFAULT NULL,  
    sync_option      IN VARCHAR2 DEFAULT 'SYNC'  
);  
```

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>synced_objects</td>
<td>A JSON object specifying the objects to be synced, as described in the <code>RUN_SYNC</code> procedure.</td>
</tr>
<tr>
<td>error_semantics</td>
<td>(Optional) Error behavior, as specified for <code>RUN_SYNC</code>. Default is <code>SKIP_ERRORS</code>.</td>
</tr>
</tbody>
</table>
## DBMS_DCAT Package

### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>log_level</td>
<td>(Optional) Logging level, as specified for <code>RUN_SYNC</code>. Default is <code>INFO</code>.</td>
</tr>
<tr>
<td>repeat_interval</td>
<td>Repeat interval for the job, with the same semantics as the repeat interval parameter of the <code>DBMS_SCHEDULER.CREATE_JOB</code> procedure. For details on the <code>repeat_interval</code>, see Overview of Creating Jobs.</td>
</tr>
<tr>
<td>force</td>
<td>(Optional) If <code>TRUE</code>, existing sync jobs are deleted first. If <code>FALSE</code>, the <code>CREATE_SYNC_JOB</code> procedure fails if a sync job already exists. Default is <code>FALSE</code>.</td>
</tr>
<tr>
<td>grant_read</td>
<td>(Optional) List of users/roles to be granted READ on the synced external tables, as described for procedure <code>RUN_SYNC</code>. See <code>DBMS_DCAT.RUN_SYNC</code> Procedure.</td>
</tr>
<tr>
<td>sync_option</td>
<td>(Optional) Behavior with respect to entities that have already been synced through a previous <code>RUN_SYNC</code> operation, as described for procedure <code>RUN_SYNC</code>. See <code>DBMS_DCAT.RUN_SYNC</code> Procedure.</td>
</tr>
</tbody>
</table>

### DROP_SYNC_JOB Procedure

This procedure drops an existing sync job.

**Syntax**

```sql
PROCEDURE DBMS_DCAT.DROP_SYNC_JOB;
```

**Parameters**

None.

### DROP_SYNCED_SCHEMAS Procedure

This procedure drops all previously synchronized schemas.

**Syntax**

```sql
PROCEDURE DBMS_DCAT.DROP_SYNCED_SCHEMAS;
```

**Parameters**

None.

### Data Catalog Views

Data Catalog integration with Autonomous Database provides numerous tables and views. These tables and views help you understand:

- Available Data Catalog assets. Get information about any type of Data Catalog asset - including databases, object stores, and more.
- Information about the Data Catalog Object Storage assets and entities that have been synchronized with Autonomous Database. This includes details about how Data Catalog...

---

**ORACLE**

A-131
items (assets, folders and entities) map to Autonomous Database objects (i.e. schemas and external tables).

- Metadata sync executions. Review details about sync jobs, including any issues that may have occurred during synchronization.

**Topics**

- ALL_DCAT_GLOBAL_ACCESSIBLE_CATALOGS View
- ALL_DCAT_LOCAL_ACCESSIBLE_CATALOGS View
- ALL_DCAT_GLOBAL_ACCESSIBLE_CATALOGS View
- ALL_DCAT_LOCAL_ACCESSIBLE_CATALOGS View
- ALL_DCAT_ASSETS View
- ALL_DCAT_FOLDERS View
- ALL_DCAT_ENTITIES View
- ALL_DCAT_ATTRIBUTES View
- ALL_DCAT_CONNECTIONS View
- DCAT_ENTITIES View
- DCAT_ATTRIBUTES View
- DBMS_DCAT$SYNC_LOG View

**ALL_DCAT_GLOBAL_ACCESSIBLE_CATALOGS View**

This view lists all accessible catalogs across all regions, along with the level of access privileges for each catalog.

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOG_ID</td>
<td>VARCHAR2 (4000)</td>
<td>Catalog OCID</td>
</tr>
<tr>
<td>CATALOG_NAME</td>
<td>VARCHAR2 (4000)</td>
<td>Name of the catalog</td>
</tr>
<tr>
<td>CATALOG_REGION</td>
<td>VARCHAR2 (4000)</td>
<td>Name of the catalog region</td>
</tr>
<tr>
<td>CATALOG_SCORE</td>
<td>NUMBER</td>
<td>The catalog score is a numeric value calculated from the privileges configured for the Data Catalog access credential. A higher catalog score means greater privileges, which may equate to a higher likelihood that this catalog is intended for use with this Autonomous Database instance.</td>
</tr>
</tbody>
</table>

**ALL_DCAT_LOCAL_ACCESSIBLE_CATALOGS View**

This view lists all accessible catalogs in the current region, along with the level of access privileges for each catalog.

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATALOG_ID</td>
<td>VARCHAR2 (4000)</td>
<td>Catalog OCID</td>
</tr>
<tr>
<td>CATALOG_NAME</td>
<td>VARCHAR2 (4000)</td>
<td>Name of the catalog</td>
</tr>
</tbody>
</table>
### ALL_DCAT_ASSETS View

The Data Catalog assets that this database is authorized to access.

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY</td>
<td>VARCHAR2(4000)</td>
<td>Asset key</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td>VARCHAR2(4000)</td>
<td>Asset display name</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR2(4000)</td>
<td>Asset description</td>
</tr>
<tr>
<td>CATALOG_ID</td>
<td>VARCHAR2(4000)</td>
<td>OCID for the Data Catalog containing the asset</td>
</tr>
<tr>
<td>EXTERNAL_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Base Object Storage URI for the asset</td>
</tr>
<tr>
<td>URI</td>
<td>VARCHAR2(4000)</td>
<td>Asset URI for the Data Catalog API</td>
</tr>
<tr>
<td>TIME_CREATED</td>
<td>TIMESTAMP(6) WITH TIMEZONE</td>
<td>The date and time the data asset was created</td>
</tr>
<tr>
<td>TYPE_KEY</td>
<td>VARCHAR2(4000)</td>
<td>The key of the data asset type (currently, only Object Storage data assets are supported). Type keys can be found via the /types Data Catalog endpoint.</td>
</tr>
<tr>
<td>LIFECYCLE_STATE</td>
<td>VARCHAR2(4000)</td>
<td>The current state of the data asset. For more information on possible life cycle states, see the Data Catalog DataAsset Reference for a list of possible states for lifecycleState.</td>
</tr>
</tbody>
</table>

### ALL_DCAT_FOLDERS View

Metadata for the Object Storage buckets containing the data files for the Logical Entities.

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY</td>
<td>VARCHAR2(4000)</td>
<td>Folder key</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td>VARCHAR2(4000)</td>
<td>Folder display name</td>
</tr>
<tr>
<td>BUSINESS_NAME</td>
<td>VARCHAR2(4000)</td>
<td>Folder business name</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR2(4000)</td>
<td>Folder description</td>
</tr>
<tr>
<td>DATA_ASSET_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Key for the data asset containing the folder</td>
</tr>
<tr>
<td>PARENT_FOLDER_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Key for the parent folder (currently, this is the data asset key)</td>
</tr>
<tr>
<td>PATH</td>
<td>VARCHAR2(4000)</td>
<td>Full path for the folder</td>
</tr>
<tr>
<td>EXTERNAL_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Object Storage URI for the bucket</td>
</tr>
</tbody>
</table>
### Column | Datatype | Description
--- | --- | ---
TIME_EXTERNAL | TIMESTAMP(6) WITH TIMEZONE | The last modified timestamp of this folder
TIME_CREATED | TIMESTAMP(6) WITH TIMEZONE | The date/time the folder was created
URI | VARCHAR2(4000) | URI to the folder instance in the Data Catalog API.
LIFECYCLE_STATE | VARCHAR2(4000) | The current state of the folder. For more information on possible life cycle states, see the Data Catalog Folder Reference for a list of possible states for lifecycleState.

### ALL_DCAT_ENTITIES View

The Data Catalog logical entities this database is authorized to access.

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY</td>
<td>VARCHAR2(4000)</td>
<td>Entity key</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td>VARCHAR2(4000)</td>
<td>Entity display name</td>
</tr>
<tr>
<td>BUSINESS_NAME</td>
<td>VARCHAR2(4000)</td>
<td>Entity business name</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR2(4000)</td>
<td>Logical entity description</td>
</tr>
<tr>
<td>DATA_ASSET_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Asset key</td>
</tr>
<tr>
<td>FOLDER_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Folder unique key</td>
</tr>
<tr>
<td>FOLDER_NAME</td>
<td>VARCHAR2(4000)</td>
<td>Folder name (bucket)</td>
</tr>
<tr>
<td>EXTERNAL_KEY</td>
<td>VARCHAR2(4000)</td>
<td>External key for the logical entity</td>
</tr>
<tr>
<td>PATTERN_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Key of the associated pattern for the logical entity</td>
</tr>
<tr>
<td>REALIZED_EXPRESSION</td>
<td>VARCHAR2(4000)</td>
<td>The regular expression used to obtain the files for this logical entity</td>
</tr>
<tr>
<td>PATH</td>
<td>VARCHAR2(4000)</td>
<td>Full path for the logical entity</td>
</tr>
<tr>
<td>TIME_CREATED</td>
<td>TIMESTAMP(6) WITH TIMEZONE</td>
<td>Date and time the entity was created</td>
</tr>
<tr>
<td>TIME_UPDATED</td>
<td>TIMESTAMP(6) WITH TIMEZONE</td>
<td>Last time a change was made to the data entity</td>
</tr>
<tr>
<td>UPDATED_BY_ID</td>
<td>VARCHAR2(4000)</td>
<td>OCID of the user who updated this object in the Data Catalog</td>
</tr>
<tr>
<td>URI</td>
<td>VARCHAR2(4000)</td>
<td>URI of the entity instance in the API</td>
</tr>
<tr>
<td>LIFECYCLE_STATE</td>
<td>VARCHAR2(4000)</td>
<td>The current state of the entity. For more information on possible life cycle states, see the Data Catalog Entity Reference for a list of possible states for lifecycleState.</td>
</tr>
</tbody>
</table>

### ALL_DCAT_ATTRIBUTES View

The Data Catalog attributes this database is authorized to access.
<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY</td>
<td>NUMBER</td>
<td>Attribute key</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td>VARCHAR2(4000)</td>
<td>Attribute display name</td>
</tr>
<tr>
<td>BUSINESS_NAME</td>
<td>VARCHAR2(4000)</td>
<td>Attribute business name</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR2(4000)</td>
<td>Attribute description</td>
</tr>
<tr>
<td>DATA_ASSET_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Data asset key</td>
</tr>
<tr>
<td>FOLDER_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Folder key</td>
</tr>
<tr>
<td>ENTITY_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Entity key</td>
</tr>
<tr>
<td>EXTERNAL_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Unique external key for the attribute</td>
</tr>
<tr>
<td>LENGTH</td>
<td>NUMBER</td>
<td>Maximum external allowed length of the attribute</td>
</tr>
<tr>
<td>PRECISION</td>
<td>NUMBER</td>
<td>Precision of the attribute value (usually applies to float data type)</td>
</tr>
<tr>
<td>SCALE</td>
<td>NUMBER</td>
<td>Scale of the attribute value (usually applies to float data type)</td>
</tr>
<tr>
<td>IS_NULLABLE</td>
<td>NUMBER</td>
<td>Identifies if this attribute can be assigned null values</td>
</tr>
<tr>
<td>URI</td>
<td>VARCHAR2(4000)</td>
<td>URI to the attribute instance in the Data Catalog API</td>
</tr>
<tr>
<td>LIFECYCLE_STATE</td>
<td>VARCHAR2(4000)</td>
<td>The current state of the attribute. For more information on possible life cycle states, see the Data Catalog Attribute Reference for a list of possible states for lifecycleState.</td>
</tr>
<tr>
<td>TIME_CREATED</td>
<td>TIMESTAMP(6) WITH TIME ZONE</td>
<td>The date and time the attribute was created</td>
</tr>
<tr>
<td>EXTERNAL_DATA_TYPE</td>
<td>VARCHAR2(4000)</td>
<td>Data type of the attribute as defined in the external system</td>
</tr>
<tr>
<td>MIN_COLLECTION_COUNT</td>
<td>NUMBER</td>
<td>Minimum number of elements, if the type of the attribute is a collection type</td>
</tr>
<tr>
<td>MAX_COLLECTION_COUNT</td>
<td>NUMBER</td>
<td>Maximum number of elements, if the type of the attribute is a collection type</td>
</tr>
<tr>
<td>DATATYPE_ENTITY_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Entity key that represents the datatype of this attribute, applicable if this attribute is a complex type</td>
</tr>
<tr>
<td>EXTERNAL_DATATYPE_ENTITY_KEY</td>
<td>VARCHAR2(4000)</td>
<td>External entity key that represents the datatype of this attribute, applicable if this attribute is a complex type</td>
</tr>
<tr>
<td>PARENT_ATTRIBUTE_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Attribute key that represents the parent attribute of this attribute, applicable if the parent attribute is of complex datatype</td>
</tr>
<tr>
<td>EXTERNAL_PARENT_ATTRIBUTE_KEY</td>
<td>VARCHAR2(4000)</td>
<td>External attribute key that represents the parent attribute of this attribute, applicable if the parent attribute is of complex datatype</td>
</tr>
<tr>
<td>PATH</td>
<td>VARCHAR2(4000)</td>
<td>Full path of the attribute</td>
</tr>
</tbody>
</table>

**ALL_DCAT_CONNECTIONS View**

A view that contains information about the data catalog(s) connected to this instance.
<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPARTMENT_ID</td>
<td>VARCHAR2(4000)</td>
<td>OCID for the compartment where the Data Catalog instance resides</td>
</tr>
<tr>
<td>INSTANCE_ID</td>
<td>VARCHAR2(4000)</td>
<td>OCID for the Data Catalog instance</td>
</tr>
<tr>
<td>REGION</td>
<td>VARCHAR2(4000)</td>
<td>Region for the Data Catalog instance</td>
</tr>
<tr>
<td>ENDPOINT</td>
<td>VARCHAR2(4000)</td>
<td>Endpoint for the Data Catalog instance</td>
</tr>
<tr>
<td>CREATED</td>
<td>TIMESTAMP</td>
<td>When the Data Catalog instance was created</td>
</tr>
<tr>
<td>NAME</td>
<td>VARCHAR2(4000)</td>
<td>Name of the Data Catalog instance</td>
</tr>
<tr>
<td>LAST_UPDATED</td>
<td>TIMESTAMP</td>
<td>Timestamp of the last update of the connection to the Data Catalog instance</td>
</tr>
<tr>
<td>LATEST_OPERATION_ID</td>
<td>NUMBER</td>
<td>The id of the last synchronization operation</td>
</tr>
<tr>
<td>DATA_CATALOG_CREDENTIAL</td>
<td>VARCHAR2(128)</td>
<td>Credential used for accessing the Data Catalog</td>
</tr>
<tr>
<td>OBJECT_STORE_CREDENTIAL</td>
<td>VARCHAR2(128)</td>
<td>Credential used by the external table driver for accessing the Object Store</td>
</tr>
</tbody>
</table>

**DCAT_ENTITIES View**

Describes the mapping of logical entities to external tables.

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSET_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Data Catalog asset key</td>
</tr>
<tr>
<td>ENTITY_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Data Catalog entity key</td>
</tr>
<tr>
<td>FOLDER_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Data Catalog folder key</td>
</tr>
<tr>
<td>ORACLE_TABLE_NAME</td>
<td>VARCHAR2(128)</td>
<td>Mapped table name</td>
</tr>
<tr>
<td>ORACLE_SCHEMA_NAME</td>
<td>VARCHAR2(128)</td>
<td>Mapped schema name</td>
</tr>
<tr>
<td>ENTITY_ORACLE_DB_SCHEMA</td>
<td>VARCHAR2(4000)</td>
<td>The entity's oracle-db-schema custom property used to derive the schema</td>
</tr>
<tr>
<td>ASSET_ORACLE_DB_SCHEMA</td>
<td>VARCHAR2(4000)</td>
<td>The data asset's oracle-db-schema custom property used to derive the schema</td>
</tr>
<tr>
<td>FOLDER_ORACLE_DB_SCHEMA</td>
<td>VARCHAR2(4000)</td>
<td>The folder's oracle-db-schema custom property used to derive the schema</td>
</tr>
</tbody>
</table>

**DCAT_ATTRIBUTES View**

Lists the mapping of logical entity attributes to external table columns.

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSET_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Data Catalog asset key</td>
</tr>
<tr>
<td>ENTITY_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Data Catalog entity key</td>
</tr>
<tr>
<td>ATTRIBUTE_KEY</td>
<td>VARCHAR2(4000)</td>
<td>Data Catalog attribute key</td>
</tr>
<tr>
<td>ORACLE_COLUMN_NAME</td>
<td>VARCHAR2(128)</td>
<td>Mapped column name</td>
</tr>
</tbody>
</table>
The DBMS_DCAT$SYNC_LOG view provides easy access to the log table for the last sync operation executed by the current user.

Every call to the RUN_SYNC procedure is logged to a new log table, pointed to by the LOGFILE_TABLE field of USER_LOAD_OPERATIONS. The log tables are automatically dropped after 2 days, and users can clear all sync logs using the DBMS_CLOUD.DELETE_ALL_OPERATIONS procedure where type is DCAT_SYNC.

The DBMS_DCAT$SYNC_LOG view automatically identifies the latest log table. The schema for the DBMS_DCAT$SYNC_LOG view is described below and the access permissions are identical to those of the individual log tables. By default READ is granted to the dbms_dcat role and to the ADMIN user.

The log tables have the following format:

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Timestamp for the log entry.</td>
</tr>
<tr>
<td>LOG_LEVEL</td>
<td>VARCHAR2(32)</td>
<td>The entry log level can have one of the following values: OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE, ALL.</td>
</tr>
<tr>
<td>LOG_DETAILS</td>
<td>VARCHAR2(32767)</td>
<td>The log message.</td>
</tr>
</tbody>
</table>

The log tables have the following format:

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG_TIMESTAMP</td>
<td>TIMESTAMP</td>
<td>Timestamp for the log entry.</td>
</tr>
<tr>
<td>LOG_LEVEL</td>
<td>VARCHAR2(32)</td>
<td>The entry log level can have one of the following values: OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE, ALL.</td>
</tr>
<tr>
<td>LOG_DETAILS</td>
<td>VARCHAR2(32767)</td>
<td>The log message.</td>
</tr>
</tbody>
</table>

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

```sql
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'));
```

Type | Object Owner | Object Name | Column Name | Reason
--- | ------------ | ----------- | ----------- |  
|     | ADMIN        | SALES       | CUST_NOTES  | Physical column exceeds STANDARD length limit

1 rows selected.

MODIFY_MAX_STRING_SIZE Procedure

This procedure updates the value of the MAX_STRING_SIZE parameter to a given value.

Syntax

```
DBMS_MAX_STRING_SIZE.MODIFY_MAX_STRING_SIZE(
    new_value    IN VARCHAR2);
```

Where: user_account is the user account name (schema name).

Parameters

<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

SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME = 'max_string_size';

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_string_size</td>
<td>EXTENDED</td>
</tr>
</tbody>
</table>

```
BEGIN
    DBMS_MAX_STRING_SIZE.MODIFY_MAX_STRING_SIZE('STANDARD');
END;
/
```

PL/SQL procedure successfully completed.

SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME = 'max_string_size';

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_string_size</td>
<td>STANDARD</td>
</tr>
</tbody>
</table>

**DBMS_AUTO_PARTITION** Package

The DBMS_AUTO_PARTITION package provides administrative routines for managing automatic partitioning of schemas and tables.

**Topics**

- **CONFIGURE Procedure**
- **VALIDATE_CANDIDATE_TABLE** Function
- **RECOMMEND_PARTITION_METHOD** Function
- **APPLY_RECOMMENDATION** Procedure
- **REPORT_ACTIVITY** Function
- **REPORT_LAST_ACTIVITY** Function

**CONFIGURE Procedure**

This procedure configures settings for automatic partitioning in Autonomous Database.

**Syntax**

```sql
DBMS_AUTO_PARTITION.CONFIGURE (
    PARAMETER_NAME  IN VARCHAR2,
    PARAMETER_VALUE IN VARCHAR2,
    ALLOW           IN BOOLEAN DEFAULT TRUE);
```
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PARAMETER_NAME</td>
<td>Name of the automatic partitioning configuration parameter to update. It can have one of the following values:</td>
</tr>
<tr>
<td></td>
<td>• AUTO_PARTITION_MODE</td>
</tr>
<tr>
<td></td>
<td>• AUTO_PARTITION_SCHEMA</td>
</tr>
<tr>
<td></td>
<td>• AUTO_PARTITION_TABLE</td>
</tr>
<tr>
<td></td>
<td>• AUTO_PARTITION_REPORT_RETENTION</td>
</tr>
<tr>
<td>AUTO_PARTITION_MODE</td>
<td>sets the mode of automatic partitioning operation, and has one of the following values:</td>
</tr>
<tr>
<td></td>
<td>• IMPLEMENT: In this mode, automatic partitioning generates a report and modifies the existing table using the recommended partition method.</td>
</tr>
<tr>
<td></td>
<td>• REPORT ONLY: In this mode, automatic partitioning generates a report but existing tables are not modified. This is the default value.</td>
</tr>
<tr>
<td></td>
<td>• OFF: In this mode, automatic partitioning is prevented from generating, considering, or applying recommendations. It does not disable existing automatic partitioned tables.</td>
</tr>
<tr>
<td>AUTO_PARTITION_SCHEMA</td>
<td>sets schemas to include or exclude from using automatic partitioning. Its behavior is controlled by the allow parameter. The automatic partitioning process manages two schema lists.</td>
</tr>
<tr>
<td></td>
<td>1. Inclusion list is the list of schemas, case-sensitive, that can use automatic partitioning.</td>
</tr>
<tr>
<td></td>
<td>2. Exclusion list is the list of schemas, case-sensitive, that cannot use automatic partitioning.</td>
</tr>
<tr>
<td>AUTO_PARTITION_TABLE</td>
<td>sets tables to include or exclude from using auto partitioning. The parameter value is &lt;schema_name&gt;.&lt;table_name&gt;. The automatic partitioning process manages two table lists.</td>
</tr>
<tr>
<td></td>
<td>1. Inclusion list is the list of tables, case-sensitive, that can use automatic partitioning.</td>
</tr>
<tr>
<td></td>
<td>2. Exclusion list is the list of tables, case-sensitive, that cannot use automatic partitioning.</td>
</tr>
</tbody>
</table>

Initially, both lists are empty, and all schemas in the database can use automatic partitioning. If the inclusion list contains one or more schemas, then only the schemas listed in the inclusion list can use automatic partitioning. If the inclusion list is empty and the exclusion list contains one or more schemas, then all schemas use automatic partitioning except the schemas listed in the exclusion list. If both lists contain one or more schemas, then all schemas use automatic partitioning except the schemas listed in the exclusion list.

AUTO_PARTITION_TABLE sets tables to include or exclude from using auto partitioning. The parameter value is <schema_name>.<table_name>. The automatic partitioning process manages two table lists.

1. Inclusion list is the list of tables, case-sensitive, that can use automatic partitioning.
2. Exclusion list is the list of tables, case-sensitive, that cannot use automatic partitioning.

Initially, both lists are empty, and all tables in the database can use automatic partitioning. If the inclusion list contains one or more tables, then only the tables listed in the inclusion list can use automatic partitioning. If the inclusion list is empty and the exclusion list contains one or more tables, then all tables use automatic partitioning except the tables listed in the exclusion list. If both lists contain one or more tables, then all tables use automatic partitioning except the tables listed in the exclusion list. If a table is not on either list, the schema inclusion and exclusion lists decide if a...
**Parameter** | **Description**  
---|---  
| table is a candidate table for automatic partitioning. If there is a conflict between the schema level lists and the table level lists, the table level lists take precedence. To remove all tables from inclusion and exclusion lists run:  
| DBMS_AUTO_PARTITION.CONFIGURE('AUTO_PARTITION_TABLE', NULL);  
| AUTO_PARTITION_REPORT_RETENTION sets the number of days for which automatic partitioning logs are retained in the database before they are deleted. An automatic partitioning report cannot be generated for a period beyond the value specified for this value. Default value is 90 days.  
| PARAMETER_VALUE | Value for the configuration setting specified in parameter_name. When set to NULL, the configuration setting is assigned its default value.  
| ALLOW | Applicable only for the AUTO_PARTITION_SCHEMA or AUTO_PARTITION_TABLE configuration settings with one of the following values:  
| TRUE | adds specified schema or table to the inclusion list.  
| FALSE | removes specified schema or table from the exclusion list.  
| NULL | removes specified schema or table from the list to which currently assigned.  
| Refer to the description of the AUTO_PARTITION_SCHEMA and AUTO_PARTITION_TABLE configuration settings for more information about inclusion lists and exclusion lists.  

**Usage Notes**

- You can check the current setting for automatic partitioning configuration using the following SQL:

```sql
SELECT * FROM DBA_AUTO_PARTITION_CONFIG;
```

- Unlike automatic indexing, automatic partitioning does not run periodically as a background task. Automatic partitioning only runs when you invoke it using the DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD function.

**VALIDATE_CANDIDATE_TABLE Function**

This function checks if the given table is a valid candidate for automatic partitioning in Autonomous Database.

**Valid Candidate**

To be a valid candidate, the following tests must pass:

- Table passes inclusion and exclusion tests specified by AUTO_PARTITION_SCHEMA and AUTO_PARTITION_TABLE configuration parameters.
- Table exists and has up-to-date statistics.
• Table is at least 64 GB.
• Table has 5 or more queries in the SQL tuning set that scanned the table.
• Table does not contain a LONG data type column.
• Table is not manually partitioned.
• Table is not an external table, an internal/external hybrid table, a temporary table, an index-organized table, or a clustered table.
• Table does not have a domain index or bitmap join index.
• Table is not an advance queuing, materialized view, or flashback archive storage table.
• Table does not have nested tables, or certain other object features.

Returns:
• VALID if the table is a valid candidate for autonomous partitioning
• INVALID: <reason> if the table is not a valid candidate for autonomous partitioning, and <reason> is a string describing why the table is not a valid candidate.

Syntax

```
DBMS_AUTO_PARTITION.VALIDATE_CANDIDATE_TABLE (sqlset_owner IN VARCHAR2 DEFAULT 'SYS',
        sqlset_name IN VARCHAR2 DEFAULT 'SYS_AUTO_STS',
        table_owner IN VARCHAR2,
        table_name IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLSET_OWNER,</td>
<td>Name of SQL tuning set representing the workload to be evaluated.</td>
</tr>
<tr>
<td>SQLSET_NAME</td>
<td></td>
</tr>
<tr>
<td>TABLE_OWNER,</td>
<td>Name of a table to validate as a candidate for automatic partitioning.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td></td>
</tr>
</tbody>
</table>

Usage Notes

• As an example, you can check the validity of a sample table, LINEORDER in schema TEST, with the following SQL:

```
SELECT DBMS_AUTO_PARTITION.VALIDATE_CANDIDATE_TABLE
    ( table_owner => 'TEST',
      table_name => 'LINEORDER')
FROM DUAL;
```
RECOMMEND_PARTITION_METHOD Function

This function returns a recommendation ID that can be used with APPLY_RECOMMENDATION procedure to apply the recommendation, or can be used with DBA_AUTO_PARTITION_RECOMMENDATIONS view to retrieve details of the recommendations for automatic partitioning in Autonomous Database.

Syntax

```sql
DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD
( SQLSET_OWNER     IN VARCHAR2   DEFAULT 'SYS',
  SQLSET_NAME      IN VARCHAR2   DEFAULT 'SYS_AUTO_STS',
  TABLE_OWNER      IN VARCHAR2   DEFAULT NULL,
  TABLE_NAME       IN VARCHAR2   DEFAULT NULL,
  TIME_LIMIT       IN INTERVAL DAY TO SECOND DEFAULT INTERVAL '1' DAY,
  REPORT_TYPE      IN VARCHAR2   DEFAULT 'TEXT',
  REPORT_SECTION   IN VARCHAR2   DEFAULT 'SUMMARY',
  REPORT_LEVEL     IN VARCHAR2   DEFAULT 'TYPICAL')
RETURN RAW;
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLSET_OWNER,</td>
<td>Name of SQL tuning set representing the workload to be evaluated.</td>
</tr>
<tr>
<td>SQLSET_NAME</td>
<td></td>
</tr>
<tr>
<td>TABLE_OWNER,</td>
<td>Name of a table to validate as a candidate for automatic partitioning.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td></td>
</tr>
<tr>
<td>TIME_LIMIT</td>
<td>When the function chooses the tables for which to generate recommendations,</td>
</tr>
<tr>
<td></td>
<td>Table_OWNER and TABLE_NAME are NULL}, parameter limits how long the function</td>
</tr>
<tr>
<td></td>
<td>runs before it stops looking for new candidate tables to partition. Once</td>
</tr>
<tr>
<td></td>
<td>started processing a table, process will not terminate. It is expected that</td>
</tr>
<tr>
<td></td>
<td>the function may run longer than this parameter. If this parameter is NULL</td>
</tr>
<tr>
<td></td>
<td>there is no time limit. The default is 1 day.</td>
</tr>
<tr>
<td>REPORT_TYPE</td>
<td>Used to generate report for recommended partition method. See REPORT_ACTIVITY</td>
</tr>
<tr>
<td></td>
<td>Function for details.</td>
</tr>
<tr>
<td>REPORT_SECTION</td>
<td>Used to generate persistent report for recommended partition method.</td>
</tr>
<tr>
<td></td>
<td>See REPORT_ACTIVITY Function for details.</td>
</tr>
<tr>
<td>REPORT_LEVEL</td>
<td>Used to generate report for recommended partition method. See REPORT_ACTIVITY</td>
</tr>
<tr>
<td></td>
<td>Function for details.</td>
</tr>
</tbody>
</table>

Usage Notes

- The AUTO_PARTITION_MODE controls the actions taken by this function:
  - IMPLEMENT: In this mode, automatic partitioning generates a report and modifies the existing table using the recommended partition method.
  - REPORT ONLY: In this mode, automatic partitioning generates a report generated but existing tables are not modified. This is the default value.
In this mode, automatic partitioning prevented from producing, considering, or applying new recommendations. It does not disable existing automatic partitioned tables.

- Unlike automatic indexing, automatic partitioning does not run periodically as a background task. Automatic partitioning only runs when you invoke it using the `DBMS_AUTO_PARTITION.RECOMMEND_PARTITION_METHOD` function.

Return Values

This function returns a recommendation ID that can be used as follows:

- Use with `DBMS_AUTO_PARTITION.APPLY_RECOMMENDATION` to apply the recommendation.
- Use with `DBA_AUTO_PARTITION_RECOMMENDATIONS` view to retrieve details of the recommendations. For example:

  ```sql
  SELECT PARTITION_METHOD, PARTITION_KEY
  FROM DBA_AUTO_PARTITION_RECOMMENDATIONS
  WHERE RECOMMENDATION_ID = :RECOMMENDATION_ID;
  ```

APPLY_RECOMMENDATION Procedure

This procedure applies the given recommendation in an Autonomous Database.

Syntax

```sql
DBMS_AUTO_PARTITION.APPLY_RECOMMENDATION
( RECOMMENDATION_ID  IN RAW,
  TABLE_OWNER        IN VARCHAR2   DEFAULT NULL,
  TABLE_NAME         IN VARCHAR2   DEFAULT NULL);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RECOMMENDATION_ID</td>
<td>Recommendation ID returned from <code>RECOMMEND_PARTITION_METHOD</code> function or queried from <code>DBA_AUTO_PARTITION_RECOMMENDATIONS</code> view.</td>
</tr>
<tr>
<td>TABLE_OWNER,</td>
<td>When a single recommendation ID has recommendations for multiple tables, this optional parameter allows you to control which tables are partitioned.</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>If parameters are NULL, partition all tables recommended in the given recommendation ID.</td>
</tr>
<tr>
<td></td>
<td>If a table name is given, partition only the named table.</td>
</tr>
<tr>
<td></td>
<td>If either <code>TABLE_OWNER</code> or <code>TABLE_NAME</code> is NOT NULL, they must both be NOT NULL.</td>
</tr>
</tbody>
</table>

Usage Note:

Regardless of `AUTO_PARTITION_MODE`, this procedure raises an ORA-20000: recommendation_id was not found if either there are no accepted recommendations
associated with the `RECOMMENDATION_ID`, or all accepted recommendations associated with the `RECOMMENDATION_ID` have already been applied. The first case applies if `RECOMMENDATION_ID` was generated with `AUTO_PARTITION_MODE = OFF`. The second case applies if `RECOMMENDATION_ID` was generated with `AUTO_PARTITION_MODE = IMPLEMENT`.

**REPORT_ACTIVITY Function**

This function returns a report of the automatic partitioning operations executed during a specific period in an Autonomous Database.

**Syntax**

```sql
DBMS_AUTO_PARTITION.REPORT_ACTIVITY
( ACTIVITY_START IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
  ACTIVITY_END IN TIMESTAMP WITH TIME ZONE DEFAULT NULL,
  TYPE IN VARCHAR2 DEFAULT 'TEXT',
  SECTION IN VARCHAR2 DEFAULT 'ALL',
  LEVEL IN VARCHAR2 DEFAULT 'TYPICAL')
RETURN CLOB;
```

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTIVITY_START</td>
<td>Starting time automatic partitioning operations use for the report. If no value is specified, or <code>NULL</code> is specified, the report is generated for the last automatic partitioning operation that was executed.</td>
</tr>
<tr>
<td>ACTIVITY_END</td>
<td>Ending time automatic partitioning operations use for the report. If no value is specified, or <code>NULL</code> is specified, then the report is generated for the last automatic partitioning operation that was executed.</td>
</tr>
<tr>
<td>TYPE</td>
<td>Format of the report that has one of the following values:</td>
</tr>
<tr>
<td></td>
<td>• TEXT (default)</td>
</tr>
<tr>
<td></td>
<td>• HTML</td>
</tr>
<tr>
<td></td>
<td>• XML</td>
</tr>
<tr>
<td>SECTION</td>
<td>Sections to include in the report that has one of the following values:</td>
</tr>
<tr>
<td></td>
<td>• SUMMARY - Include only the workload summary in the report</td>
</tr>
<tr>
<td></td>
<td>• ALL - Include all the sections in the report. (default)</td>
</tr>
<tr>
<td>level</td>
<td>Level of information to include in the report that has one of the following values:</td>
</tr>
<tr>
<td></td>
<td>• TYPICAL - Include typical automatic partitioning information in the report (default).</td>
</tr>
<tr>
<td></td>
<td>• CHANGED - Include only SQL with changed performance in the report.</td>
</tr>
<tr>
<td></td>
<td>• IMPROVED - Include only SQL with improved performance in the report.</td>
</tr>
<tr>
<td></td>
<td>• REGRESSED - Include only SQL with regressed performance in the report.</td>
</tr>
<tr>
<td></td>
<td>• UNCHANGED - Include only SQL with unchanged performance in the report.</td>
</tr>
<tr>
<td></td>
<td>• ALL - Include all automatic partitioning information in the report.</td>
</tr>
</tbody>
</table>
Usage Notes

Returns: A performance analysis report for workload executed on database after recommendation is applied. This report is not stored persistently with the recommendation.

REPORT_LAST_ACTIVITY Function

This function returns a report of the most recent automatic partitioning operation executed in an Autonomous Database.

Syntax

```
DBMS_AUTO_PARTITION.REPORT_LAST_ACTIVITY
  ( TYPE    IN VARCHAR2 DEFAULT 'TEXT',
    SECTION IN VARCHAR2 DEFAULT 'ALL',
    LEVEL   IN VARCHAR2 DEFAULT 'TYPICAL')
RETURN CLOB;
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE</td>
<td>The output format of the report, see REPORT_ACTIVITY Function for information.</td>
</tr>
<tr>
<td>SECTION</td>
<td>The sections included in the report, see REPORT_ACTIVITY Function for information.</td>
</tr>
<tr>
<td>LEVEL</td>
<td>The level of information included in the report, see REPORT_ACTIVITY Function for information.</td>
</tr>
</tbody>
</table>

Usage Notes

Returns: A performance analysis report for workload executed on database after latest recommendation is applied. This report is not stored persistently with the recommendation.

CS_RESOURCE_MANAGER Package

The CS_RESOURCE_MANAGER package provides an interface to list and update consumer group parameters, and to revert parameters to default values.

Topics

- LIST_CURRENT_RULES Function
- LIST_DEFAULT_RULES Function
- REVERT_TO_DEFAULT_VALUES Procedure
- UPDATE_PLAN_DIRECTIVE Procedure
LIST_CURRENT_RULES Function

This function lists the parameter values for each consumer group.

Syntax

```sql
CS_RESOURCE_MANAGER.LIST_CURRENT_RULES
RETURN TABLE;
```

Example

```sql
SELECT * FROM CS_RESOURCE_MANAGER.LIST_CURRENT_RULES();
```

<table>
<thead>
<tr>
<th>CONSUMER_GROUP</th>
<th>ELAPSED_TIME_LIMIT</th>
<th>IO_MEGABYTES_LIMIT</th>
<th>SHARES</th>
<th>CONCURRENCY_LIMIT</th>
<th>DEGREE_OF_PARALLELISM</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIGH</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>MEDIUM</td>
<td>2</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LOW</td>
<td>900</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

LIST_DEFAULT_RULES Function

This function returns the default values for all consumer groups.

Syntax

```sql
CS_RESOURCE_MANAGER.LIST_DEFAULT_RULES
RETURN TABLE;
```

Usage Note

- By default the parallel degree policy value is **MANUAL** for the **TPURGENT** consumer group. The `CS_RESOURCE_MANAGER.LIST_DEFAULT_RULES` function shows no value for the default value for the **DEGREE_OF_PARALLELISM** for the **TPURGENT** consumer group.

Example

```sql
SELECT * FROM CS_RESOURCE_MANAGER.LIST_DEFAULT_RULES();
```

<table>
<thead>
<tr>
<th>CONSUMER_GROUP</th>
<th>ELAPSED_TIME_LIMIT</th>
<th>IO_MEGABYTES_LIMIT</th>
<th>SHARES</th>
<th>CONCURRENCY_LIMIT</th>
<th>DEGREE_OF_PARALLELISM</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIGH</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>MEDIUM</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>LOW</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
REVERT_TO_DEFAULT_VALUES Procedure

This procedure reverts the specified resource manager's plan properties to default values.

Syntax

```
CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(
    consumer_group   IN VARCHAR2,
    shares           IN BOOLEAN   DEFAULT FALSE,
    concurrency_limit IN BOOLEAN   DEFAULT FALSE);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>consumer_group</td>
<td>Specifies the consumer group to revert. Depend on the workload, valid values are: HIGH, MEDIUM, LOW, TP, or TPURGENT.</td>
</tr>
<tr>
<td>shares</td>
<td>When the value is TRUE, revert shares for the service to the default value.</td>
</tr>
<tr>
<td>concurrency_limit</td>
<td>When the value is TRUE, revert the concurrency_limit for the service to the default value. When you revert the concurrency_limit, both the concurrency_limit and the degree_of_parallelism values are set to their default values.</td>
</tr>
</tbody>
</table>

Usage Note

- When the workload type is Data Warehouse, the valid values for consumer_group are HIGH, MEDIUM, or LOW.

Examples

```
BEGIN
    CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(
        consumer_group => 'MEDIUM',
        concurrency_limit => TRUE);
END;
/
```

```
BEGIN
    CS_RESOURCE_MANAGER.REVERT_TO_DEFAULT_VALUES(
        consumer_group => 'HIGH',
        shares => TRUE);
END;
/
```
UPDATE_PLAN_DIRECTIVE Procedure

Use this procedure to update the resource plan for a specified consumer group.

Syntax

```
CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(
    consumer_group         IN VARCHAR2,
    io_megabytes_limit     IN NUMBER   DEFAULT NULL,
    elapsed_time_limit     IN NUMBER   DEFAULT NULL,
    shares                 IN NUMBER   DEFAULT NULL,
    concurrency_limit      IN NUMBER   DEFAULT NULL);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>consumer_group</td>
<td>Specifies the consumer group to update.</td>
</tr>
<tr>
<td>io_megabytes_limit</td>
<td>Specifies the maximum megabytes of I/O that a SQL operation can issue.</td>
</tr>
<tr>
<td>elapsed_time_limit</td>
<td>Specifies the maximum time in seconds that a SQL operation can run.</td>
</tr>
<tr>
<td>shares</td>
<td>Specifies the shares value. A higher number of shares, relative to other</td>
</tr>
<tr>
<td>concurrency_limit</td>
<td>Specifies the maximum number of concurrent SQL statements that can be</td>
</tr>
<tr>
<td></td>
<td>only valid with the MEDIUM consumer group.</td>
</tr>
</tbody>
</table>

Usage Notes

- When a SQL statement in the specified service runs more than the specified runtime limit (elapsed_time_limit) or does more I/O than the specified amount (io_megabytes_limit), then the SQL statement will be terminated.
- When the workload type is Data Warehouse, the valid values for consumer_group are HIGH, MEDIUM, or LOW.
- When the concurrency_limit parameter is specified, the only valid value for consumer_group is MEDIUM.

Examples

```
BEGIN
    CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(
        consumer_group => 'HIGH',
        shares => 8);
    CS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(
        consumer_group => 'MEDIUM',
    );
```

CS_SESSION Package

The CS_SESSION package provides an interface to switch the database service and consumer group of the existing session.

When a connection is established with an Autonomous Database, that session is assigned a consumer group. For example, a session could be created using a connection to the LOW service of an Autonomous Database. You might want to switch the consumer group, for example from LOW to HIGH. The CS_SESSION package provides an API for switching.

Consumer group affects concurrency and degree of parallelism (DOP). For example, statements on a connection established to the LOW database service run serially. Statements on a connection established to the HIGH database service run in parallel. If you have a workload that requires serial statement processing with switching to a HIGH consumer group for a few statements, the CS_SESSION package enables you to switch.

- See Database Service Names for Autonomous Data Warehouse for more information.
- See Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database for more information.

SWITCH_SESSION Procedure

This procedure switches the database service and consumer group of the current session.

Syntax

```sql
CS_SESSION.SWITCH_SERVICE(service_name IN varchar2);
```
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service_name</td>
<td>Specifies the consumer group to update. Depending on the workload, valid values are: HIGH, MEDIUM, LOW, TP, or TPURGENT.</td>
</tr>
</tbody>
</table>

Usage Notes

When called, the procedure switches the session to the specified service and the related consumer group. If the specified service does not exist in that database, an error message is provided. For example, if you specify 'TP' as the service name on a data warehouse workload, the error indicates it's not a valid service name. No error is reported if the current service and the specified service are identical.

The procedure does not reset session attributes. Anything the user set for their session before calling this procedure will continue as-is. For example, if a session parameter was modified and then later the session switched to a different service, the parameter value will stay the same.

Example

```
BEGIN
    CS_SESSION.SWITCH_SERVICE('HIGH');
END;
/
```

Security and Access

The **ADMIN** user is granted **EXECUTE** privilege on **CS_SESSION** with **GRANT OPTION**. The privilege is also granted to **DWROLE** without the **GRANT OPTION**.

Additional Security Considerations

If a user is granted **EXECUTE** privileges on this procedure and you do not want that user to switch to a specific service, you can use a **AFTER SET CONTAINER** trigger to block the operation. This is achieved by creating an **AFTER SET CONTAINER trigger**.

```
CREATE OR REPLACE TRIGGER SESS_SWITCH
AFTER SET CONTAINER ON DATABASE
BEGIN
    IF SYS_CONTEXT('USERENV','SESSION_USER') = 'USER' and
       SYS_CONTEXT('USERENV','SERVICE_NAME') =
       'serviceexample_low.adwc.oraclecloud.com'
    THEN
        NULL;
    ELSE
        RAISE_APPLICATION_ERROR(-20001, 'Denied! You are not allowed to switch service in the database');
    END IF;
END;
/
```
## Error Messages

The following table describes exceptions for `CS_SESSION`.

<table>
<thead>
<tr>
<th>Error</th>
<th>Message</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>20001</td>
<td>Invalid service name. Valid values are HIGH, MEDIUM, LOW.</td>
<td>For a data warehouse workload, a value other than 'HIGH', 'MEDIUM', 'LOW' was specified.</td>
</tr>
<tr>
<td>20001</td>
<td>Invalid service name. Valid values are HIGH, MEDIUM, LOW, TP, TPURGENT.</td>
<td>For a transaction processing workload, a value other than 'HIGH', 'MEDIUM', 'LOW', 'TP', 'TPURGENT' was specified.</td>
</tr>
<tr>
<td>20002</td>
<td>Service switch failed.</td>
<td>Failed to switch to the new service.</td>
</tr>
</tbody>
</table>
Autonomous Database for Experienced Oracle Database Users

This appendix provides information on using Autonomous Database 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 Oracle Database Features in Dedicated Autonomous Database Deployments.

Topics

- About Autonomous Database for Experienced Oracle Database Users
- Autonomous Database Views
- Autonomous Database – Oracle Database Features
- Always Free Autonomous Database Oracle Database 21c Features
- Autonomous Database RMAN Recovery Catalog
- Notes for Users Migrating from Other Oracle Databases
- Database Features Unavailable in Autonomous Database

About Autonomous Database for Experienced Oracle Database Users

Autonomous Database configures and optimizes your database for you. You do not need to perform administration operations for configuring the database. SQL commands used for database administration such as `CREATE TABLESPACE` are not available. Similarly, other administrative interfaces and utilities such as `RMAN` are not available.

There are differences, depending on your workload: Data Warehouse, Transaction Processing, or JSON Database. See the following section appropriate to your workload:

- Data Warehouse Workload with Autonomous Database
- Transaction Processing and JSON Database Workloads with Autonomous Database

Data Warehouse Workload with Autonomous Database

Autonomous Database configures and optimizes your database for you, based on your workload.

Characteristics of a database with Data Warehouse workload:

- The default data and temporary tablespaces for the database are configured automatically. Adding, removing, or modifying tablespaces is not allowed. Autonomous
Database creates one tablespace or multiple tablespaces automatically depending on the storage size.

- The database character set is Unicode AL32UTF8. See Choose a Character Set for Autonomous Database for more information.
- Compression is enabled by default. Autonomous Database uses Hybrid Columnar Compression for all tables by default. You can specify different compression methods for your tables using the compression clause in your CREATE TABLE or ALTER TABLE commands.
- Oracle Database Result Cache is enabled by default for all SQL statements.

Accessing a database:

- You do not have direct access to the database node. You can create and drop directories with CREATE DIRECTORY and DROP DIRECTORY, as described in Creating and Managing Directories on Autonomous Database.
  You can use DBMS_CLOUD procedures such as DBMS_CLOUD.DELETE_FILE, DBMS_CLOUD.GET_OBJECT, and DBMS_CLOUD.PUT_OBJECT with files and objects. You do not have direct access to the local file system.

Parallel Execution with Data Warehouse workload:

- Parallelism is determined by the database service. See Database Service Names for Autonomous Data Warehouse for details for parallelism support for each database service.
- When you want to disable parallel DML operations in your session, use the following SQL command:

  ALTER SESSION DISABLE PARALLEL DML;

  See VLDB and Partitioning Guide for more information on parallel DML operations.

Manage DML Performance and Compression for Data Warehouse Workloads

Autonomous Database with Data Warehouse workloads uses Hybrid Columnar Compression for all tables by default. This gives the best compression ratio and optimal performance for direct-path load operations like the loads done using the DBMS_CLOUD package. If you perform DML operations like UPDATE and MERGE on your tables these may cause the compression ratio for the affected rows to decrease leading to larger table sizes. These operations may also perform slower compared to the same operations on an uncompressed table.

For the best compression ratio and optimal performance Oracle recommends using bulk operations like direct-path loads and CREATE TABLE AS SELECT statements. But, if your workload requires frequent DML operations like UPDATE and MERGE on large parts of a table, you can create those tables as uncompressed tables to achieve better DML performance. For example, the following statement creates the table SALES as an uncompressed table:

```sql
CREATE TABLE sales {
  prod_id     NUMBER      NOT NULL,
  cust_id     NUMBER      NOT NULL,
  time_id     DATE         NOT NULL,
};
```
At any point in time you can use the `ALTER TABLE MOVE` statement to compress these tables without impacting queries accessing them. For example, the following statement compresses the table `SALES` using Hybrid Columnar Compression.

```
ALTER TABLE sales MOVE COLUMN STORE COMPRESS FOR QUERY HIGH;
```

Create Staging Tables for Data Warehouse Workloads

Autonomous Database supports staging tables that are optimized for loading data into a data warehouse.

A staging table is a table with the `STAGING` property set. This applies the following characteristics:

- Any form of compression is explicitly turned off and disallowed on a staging table for any data load. The command `ALTER TABLE COMPRESS` is not allowed.
- Setting the `STAGING` property on an existing table does not impact the storage of existing data but does impact future data loads.
- Autonomous Database uses dynamic sampling for statistics for tables with the staging property set, and does not collect statistics on staging tables.
- Dropping staging tables immediately removes the table, bypassing the recycle bin. Setting the `recyclebin` initialization parameter to the value `ON` does not enable the recycle bin.

The characteristics of Autonomous Database partitioned staging tables includes the above, plus the following:

- Any form of compression is explicitly turned off and disallowed on all of the table's partitions and subpartitions.
- You cannot change the default attributes of the table to use compress with `ALTER TABLE MODIFY DEFAULT ATTRIBUTES`.
- You cannot perform partition maintenance operations that move data and compress the data. For example, the following are not allowed when you try to apply compression:
  
  ```
  ALTER TABLE with MOVE PARTITION, MERGE PARTITIONS, SPLIT PARTITION, or SPLIT SUBPARTITION.
  ```

- You cannot repartition a table with `ALTER TABLE MODIFY PARTITION` and specify any resulting partition to be compressed.

Define staging tables when you create a table or by altering an existing table as follows:

1. Create a table with the `STAGING` property.

   For example:

   ```
   CREATE TABLE staging_table (col1 number, col2 varchar2(100)) FOR STAGING;
   ```
CREATE TABLE part_staging_table (col1 number, col2 varchar2(100))
PARTITION BYRANGE (col1)
(PARTITION p1 VALUESLESS THAN (100),
 PARTITION pmax VALUESLESS THAN (MAXVALUE)) FOR STAGING;

2. Change an existing table to set the STAGING property.
   For example:

   ALTER TABLE staging_table FOR STAGING;

   You can alter a table to remove the STAGING property. For example:

   ALTER TABLE staging_table NOT FOR STAGING;

Note the following for altering a table with NOT FOR STAGING:

• After you alter a table with NOT FOR STAGING, the compression attribute and
  existing data are not affected and are kept as uncompressed until you explicitly
  alter the table and specify compression. You can change the table compression
  and ALTER TABLE COMPRESS is allowed.

• Altering a table with NOT FOR STAGING does not trigger statistics collection. After
  you change the table property with NOT FOR STAGING, you can collect statistics,
  either manually or automatically.

• After you alter a table with NOT FOR STAGING, when the recycle bin is enabled
  dropping the table puts the table in the recycle bin.

Transaction Processing and JSON Database Workloads with
Autonomous Database

Autonomous Database configures and optimizes your database for you, based on your
workload.

Characteristics of Autonomous Database with Transaction Processing or JSON
Database workloads:

• The default data and temporary tablespaces for the database are configured
  automatically. Adding, removing, or modifying tablespaces is not allowed.
  Autonomous Database creates one tablespace or multiple tablespaces
  automatically depending on the storage size.

• The database character set is Unicode AL32UTF8. See Choose a Character Set for
  Autonomous Database for more information.

• Compression is not enabled by default but Autonomous Database honors a
  compression clause if compression is specified on a table.

Accessing a database:

• You do not have direct access to the database node. You can create and drop
directories with CREATE DIRECTORY and DROP DIRECTORY, as described in Create
and Manage Directories.
You can use DBMS_CLOUD procedures such as DBMS_CLOUD.DELETE_FILE, DBMS_CLOUD.GET_OBJECT, and DBMS_CLOUD.PUT_OBJECT with files and objects. You do not have direct access to the local file system.

Parallel Execution with Transaction Processing or JSON Database workloads:

- Parallelism is determined by the database service you use. See Database Service Names for Autonomous Transaction Processing and Autonomous JSON Database for details for parallelism support for each database service.
- When you want to run DML operations in parallel and the database service you are using allows this, you can enable parallel DML in your session using the following SQL command:

  \[
  \text{ALTER SESSION ENABLE PARALLEL DML;}
  \]

  See VLDB and Partitioning Guide for more information on parallel DML operations.

- If you create an index manually and specify the PARALLEL clause, the PARALLEL attribute remains after the index is created. In this case SQL statements can run in parallel unbeknownst to the end user.

  To specify serial execution, change the INDEX parallel clause to NOPARALLEL or set the PARALLEL degree attribute to 1 to specify serial execution:

  \[
  \text{ALTER INDEX index_name NOPARALLEL;}
  \]

  or

  \[
  \text{ALTER INDEX index_name PARALLEL 1;}
  \]

Autonomous Database Views

Autonomous Database provides several views that are not available on Oracle Database. This topic lists the Autonomous Database specific views.

Topics

- Track Table and Partition Scan Access with Autonomous Database Views
- Track Oracle Cloud Infrastructure Resources, Cost and Usage Reports with Autonomous Database Views

Track Table and Partition Scan Access with Autonomous Database Views

Oracle Autonomous Database tracks the scan count for tables and partitions. Use the table access stats data dictionary and dynamic views to retrieve scan count information.

Topics

- GV$TABLE_ACCESS_STATS and V$TABLE_ACCESS_STATS Views
- ALL_TABLE_ACCESS_STATS and DBA_TABLE_ACCESS_ACCESS_STATS Views
- USER_TABLE_ACCESS_ACCESS_STATS View
GV$TABLE_ACCESS_STATS and V$TABLE_ACCESS_STATS Views

The GV$TABLE_ACCESS_STATS and V$TABLE_ACCESS_STATS views list the scan count for tables and partitions. The scan data collection begins at instance startup time.

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_COUNT</td>
<td>NUMBER</td>
<td>Aggregated scan count since instance startup</td>
</tr>
<tr>
<td>OBJECT_ID</td>
<td>NUMBER</td>
<td>Object ID of the table or partition</td>
</tr>
<tr>
<td>INST_ID</td>
<td>NUMBER</td>
<td>Instance number where table/partition was scanned</td>
</tr>
<tr>
<td>CON_ID</td>
<td>NUMBER</td>
<td>Container ID of the database</td>
</tr>
</tbody>
</table>

ALL_TABLE_ACCESS_STATS and DBA_TABLE_ACCESS_STATS Views

The ALL_TABLE_ACCESS_STATS and DBA_TABLE_ACCESS_STATS views list the scan count for tables and partitions. The scan data collection begins at instance startup time.

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE_OWNER</td>
<td>VARCHAR2(128)</td>
<td>Owner of the table</td>
</tr>
<tr>
<td>TABLE_NAME</td>
<td>VARCHAR2(128)</td>
<td>Name of the table</td>
</tr>
<tr>
<td>PARTITION_NAME</td>
<td>VARCHAR2(128)</td>
<td>Name of the partition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A NULL value specifies a non-partitioned table</td>
</tr>
<tr>
<td>INSTANCE_ID</td>
<td>NUMBER</td>
<td>Instance number where table or partition was scanned</td>
</tr>
<tr>
<td>READ_COUNT</td>
<td>NUMBER</td>
<td>Aggregated scan count since instance startup</td>
</tr>
</tbody>
</table>

USER_TABLE_ACCESS_STATS View

The USER_TABLE_ACCESS_STATS view lists the scan count for the user's tables and partitions. The scan data collection begins at instance startup time.

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE_NAME</td>
<td>VARCHAR2(128)</td>
<td>Name of the table</td>
</tr>
<tr>
<td>PARTITION_NAME</td>
<td>VARCHAR2(128)</td>
<td>Name of the partition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A NULL value specifies a non-partitioned table</td>
</tr>
<tr>
<td>INSTANCE_ID</td>
<td>NUMBER</td>
<td>Instance number where table/partition was scanned</td>
</tr>
<tr>
<td>READ_COUNT</td>
<td>NUMBER</td>
<td>Aggregated scan count since instance startup</td>
</tr>
</tbody>
</table>
Track Oracle Cloud Infrastructure Resources, Cost and Usage Reports with Autonomous Database Views

Oracle Autonomous Database tracks the Oracle Cloud Infrastructure resources, cost and usage reports. You can access these reports using the OCI views.

Topics

- Prerequisite Steps to Use OCI Resource Views
- OCI_AUTONOMOUSATABASES View
- OCI_BUDGET_ALERT_RULES View
- OCI_BUDGET_SUMMARY View
- OCI_COST_DATA View
- OCI_OBJECTSTORAGE_BUCKETS View
- OCI_USAGE_DATA View

Prerequisite Steps to Use OCI Resource Views

Describes the prerequisite steps you must perform to use OCI resource views on Autonomous Database.

Note:

Only ADMIN user has access to the OCI resource views by default. To access these views as another user, the ADMIN must grant READ privileges.

To query an OCI resource view, do the following:

1. Create a dynamic group that includes your Autonomous Database instance and define the required policies to access a view.

   For example, the Autonomous Database instance is specified in the resource.id parameter with an OCID:

   ```
   resource.id = '<your_Autonomous_Database_instance_OCID>'
   ```

   Each view shows the details for the policy that you must define to query the view.

   See Perform Prerequisites to Use Resource Principal with Autonomous Database for details on creating a dynamic group and defining policies.

   For example, to access all of the views, define the following policy:

   ```
   Define tenancy usage-report as
   ocid1.tenancy.oc1..aaaaaaanecd4fkpisbwjlr56u7cj631f3wffbilvqknstgtvzub7vh
   qk9qq
   Endorse dynamic-group <group-name> to read objects in tenancy usage-report
   Allow dynamic-group <group-name> to read buckets in tenancy
   ```
Allow dynamic-group <group-name> to read autonomous-database in tenancy

Allow dynamic-group <group-name> to read usage-budgets in tenancy

Note:
Do not replace the OCID in this policy with another OCID. This usage-report OCID provides the Oracle Cloud Infrastructure usage data for your tenancy.

2. Verify that resource principal is enabled for the ADMIN user on the Autonomous Database instance.

```sql
SELECT owner, credential_name FROM dba_credentials
WHERE credential_name = 'OCI$RESOURCE_PRINCIPAL' AND owner = 'ADMIN';
```

<table>
<thead>
<tr>
<th>OWNER</th>
<th>CREDENTIAL_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMIN</td>
<td>OCI$RESOURCE_PRINCIPAL</td>
</tr>
</tbody>
</table>

If the resource principal is not enabled, then enable the resource principal:

```sql
EXEC DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL();
```

See Use Resource Principal to Access Oracle Cloud Infrastructure Resources for more information.

3. Run a query on an OCI resource view.

For example:

```sql
SELECT NAME, APPROXIMATESIZE FROM OCI_OBJECTSTORAGE_BUCKETS;
SELECT * FROM OCI_USAGE_DATA;
```

OCI_AUTONOMOUS_DATABASES View

OCI_AUTONOMOUS_DATABASES describes all the Oracle Cloud Infrastructure Autonomous Databases in the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.

To query this view you need a dynamic group that includes your Autonomous Database instance and the following policy defined on that dynamic group:

Allow dynamic-group <group-name> to read autonomous-database in tenancy

This policy lets you list all Autonomous Databases in your tenancy. Optionally you can restrict it to list Autonomous Databases in a given compartment:

Allow dynamic-group <group-name> to read autonomous-database in compartment <compartment-name>
<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISPLAYNAME</td>
<td>VARCHAR2</td>
<td>The user friendly name for the Autonomous Database</td>
</tr>
<tr>
<td>REGION</td>
<td>VARCHAR2</td>
<td>Region Name</td>
</tr>
<tr>
<td>COMPARTMENTID</td>
<td>VARCHAR2</td>
<td>The OCID of the compartment</td>
</tr>
<tr>
<td>ID</td>
<td>VARCHAR2</td>
<td>The OCID of the Autonomous Database</td>
</tr>
<tr>
<td>DBNAME</td>
<td>VARCHAR2</td>
<td>The database name</td>
</tr>
<tr>
<td>LIFECYCLESTATE</td>
<td>VARCHAR2</td>
<td>The current state of the Autonomous Database</td>
</tr>
<tr>
<td>TIMECREATED</td>
<td>VARCHAR2</td>
<td>The date and time the Autonomous Database was created</td>
</tr>
<tr>
<td>DATASTORAGESIZEINTBS</td>
<td>VARCHAR2</td>
<td>The quantity of data in the database in terabytes</td>
</tr>
<tr>
<td>LICENSEMODEL</td>
<td>VARCHAR2</td>
<td>The Oracle license model that applies to the Autonomous Database</td>
</tr>
<tr>
<td>SERVICECONSOLEURL</td>
<td>VARCHAR2</td>
<td>The URL of the Service Console for the Autonomous Database</td>
</tr>
<tr>
<td>APEXDETAILS</td>
<td>CLOB</td>
<td>Information about Oracle APEX Application Development</td>
</tr>
<tr>
<td>AREPRIMARYWHITELISTEDIP</td>
<td>VARCHAR2</td>
<td>Primary White Listed IPs</td>
</tr>
<tr>
<td>SUSED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUTONOMOUSCONTAINERDATA</td>
<td>VARCHAR2</td>
<td>The Autonomous Container Database OCID</td>
</tr>
<tr>
<td>BASEID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUTONOMOUSMAINTENANCESC</td>
<td>VARCHAR2</td>
<td>Maintenance Schedule Type</td>
</tr>
<tr>
<td>NEDULETYPE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AVAILABLEUPGRADEVERSION</td>
<td>VARCHAR2</td>
<td>List of Oracle Database versions available for a database upgrade</td>
</tr>
<tr>
<td>BACKUPCONFIG</td>
<td>CLOB</td>
<td>Autonomous Database Backup Config</td>
</tr>
<tr>
<td>CONNECTIONSTRINGS</td>
<td>CLOB</td>
<td>Autonomous Database Connection Strings</td>
</tr>
<tr>
<td>CONNECTIONURLS</td>
<td>CLOB</td>
<td>Autonomous Database Connection URLs</td>
</tr>
<tr>
<td>CPUCORECOUNT</td>
<td>NUMBER</td>
<td>The number of OCPU cores to be made available to the database</td>
</tr>
<tr>
<td>CUSTOMERCONTACTS</td>
<td>CLOB</td>
<td>The Customer Contacts</td>
</tr>
<tr>
<td>DATASAFESTATUS</td>
<td>VARCHAR2</td>
<td>Status of the Data Safe registration for this Autonomous Database</td>
</tr>
<tr>
<td>DATASTORAGESIZEINGBS</td>
<td>NUMBER</td>
<td>The quantity of data in the database in gigabytes</td>
</tr>
<tr>
<td>DBVERSION</td>
<td>VARCHAR2</td>
<td>The Oracle Database version for the Autonomous Database</td>
</tr>
<tr>
<td>DATAGUARDREGIONTYPE</td>
<td>VARCHAR2</td>
<td>The Autonomous Data Guard region type of the Autonomous Database</td>
</tr>
<tr>
<td>DBWORKLOAD</td>
<td>VARCHAR2</td>
<td>The Autonomous Database workload type</td>
</tr>
<tr>
<td>DEFINEDTAGS</td>
<td>CLOB</td>
<td>Defined tags for the resource</td>
</tr>
<tr>
<td>FAILEDDATARECOVERYINSECONDS</td>
<td>NUMBER</td>
<td>Indicates the number of seconds of data loss for an Autonomous Data Guard failover</td>
</tr>
<tr>
<td>FREEFORMTAGS</td>
<td>CLOB</td>
<td>Free form tags for the resource</td>
</tr>
<tr>
<td>INFRASTRUCTURETYPE</td>
<td></td>
<td>The infrastructure type this resource belongs to</td>
</tr>
<tr>
<td>ISACCESSCONTROLENABLE</td>
<td>VARCHAR2</td>
<td>Indicates if the database level access control is enabled</td>
</tr>
<tr>
<td>ISAUTOSCALINGENABLED</td>
<td>VARCHAR2</td>
<td>Indicates if auto scaling is enabled for the Autonomous Database</td>
</tr>
<tr>
<td>ISDATAGUARDENABLED</td>
<td>VARCHAR2</td>
<td>Indicates whether the Autonomous Database has a local Autonomous Data Guard enabled</td>
</tr>
<tr>
<td>Column</td>
<td>Datatype</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ISDEDICATED</td>
<td>VARCHAR2</td>
<td>True if the database uses dedicated Exadata infrastructure</td>
</tr>
<tr>
<td>ISFREETIER</td>
<td>VARCHAR2</td>
<td>Indicates if this is an Always Free resource</td>
</tr>
<tr>
<td>ISMTLSCONNECTIONREQUIRED</td>
<td>VARCHAR2</td>
<td>Indicates whether the Autonomous Database requires mTLS connections</td>
</tr>
<tr>
<td>ISPREVIEW</td>
<td>VARCHAR2</td>
<td>Indicates if the Autonomous Database version is a preview version</td>
</tr>
<tr>
<td>ISREFRESHABLECLONE</td>
<td>VARCHAR2</td>
<td>Indicates whether the Autonomous Database is a refreshable clone</td>
</tr>
<tr>
<td>KEYHISTORYENTRY</td>
<td>CLOB</td>
<td>Key History Entry</td>
</tr>
<tr>
<td>KEYSTOREID</td>
<td>VARCHAR2</td>
<td>The OCID of the key store</td>
</tr>
<tr>
<td>KEYSTOREWALLETNAME</td>
<td>VARCHAR2</td>
<td>The wallet name for Oracle Cloud Infrastructure Vault</td>
</tr>
<tr>
<td>KMSKEYID</td>
<td>VARCHAR2</td>
<td>The OCID of the key container that is used as the master encryption key</td>
</tr>
<tr>
<td>KMSKEYLIFECYCLEDETAILS</td>
<td>VARCHAR2</td>
<td>Customer managed key lifecycle details</td>
</tr>
<tr>
<td>LIFECYCLEDETAILS</td>
<td>VARCHAR2</td>
<td>Information about the current lifecycle state</td>
</tr>
<tr>
<td>NSGIDS</td>
<td>CLOB</td>
<td>A list of the OCIDs of the network security groups</td>
</tr>
<tr>
<td>OCPUCOUNT</td>
<td>NUMBER</td>
<td>The number of OCPU cores to be made available to the database</td>
</tr>
<tr>
<td>OPENMODE</td>
<td>VARCHAR2</td>
<td>The Autonomous Database open mode</td>
</tr>
<tr>
<td>OPERATIONSINSIGHTSTATU</td>
<td>VARCHAR2</td>
<td>Status of Operations Insights for this Autonomous Database</td>
</tr>
<tr>
<td>PEERDBIDS</td>
<td>VARCHAR2</td>
<td>The list of OCIDs of standby databases located in Autonomous Data Guard</td>
</tr>
<tr>
<td>PERMISSIONLEVEL</td>
<td>CLOB</td>
<td>The Autonomous Database permission level</td>
</tr>
<tr>
<td>PRIVATEENDPOINT</td>
<td>VARCHAR2</td>
<td>The private endpoint for the resource</td>
</tr>
<tr>
<td>PRIVATEENDPOINTIP</td>
<td>VARCHAR2</td>
<td>The private endpoint IP address for the resource</td>
</tr>
<tr>
<td>PRIVATEENDPOINTLABEL</td>
<td>VARCHAR2</td>
<td>The private endpoint label for the resource</td>
</tr>
<tr>
<td>REFRESHABLEMODE</td>
<td>VARCHAR2</td>
<td>The refresh mode of the clone</td>
</tr>
<tr>
<td>REFRESHABLESTATUS</td>
<td>VARCHAR2</td>
<td>The refresh status of the clone</td>
</tr>
<tr>
<td>ROLE</td>
<td>VARCHAR2</td>
<td>The Autonomous Data Guard role</td>
</tr>
<tr>
<td>SOURCEID</td>
<td>VARCHAR2</td>
<td>The OCID of the source Autonomous Database that was cloned</td>
</tr>
<tr>
<td>SQLWEBDEVELOPERURL</td>
<td>VARCHAR2</td>
<td>The Database Actions (SQL Developer Web) URL for the Autonomous Database</td>
</tr>
<tr>
<td>STANDBYDB</td>
<td>CLOB</td>
<td>Autonomous Database Standby Summary</td>
</tr>
<tr>
<td>STANDBYWHITELISTEDIPS</td>
<td>CLOB</td>
<td>The client IP access control list</td>
</tr>
<tr>
<td>SUBNETID</td>
<td>VARCHAR2</td>
<td>The OCID of the subnet the resource is associated with</td>
</tr>
<tr>
<td>SUPPORTEDREGIONSTOCLONE</td>
<td>CLOB</td>
<td>The list of regions that support the creation of Autonomous Data Guard</td>
</tr>
<tr>
<td>SYSTEMTAGS</td>
<td>CLOB</td>
<td>System tags for this resource</td>
</tr>
<tr>
<td>TIMEDATAGUARDROLECHANGE</td>
<td>VARCHAR2</td>
<td>The date and time the Autonomous Data Guard role was switched</td>
</tr>
<tr>
<td>TIMEDELETIONOFFREEAUTONOMOUSDATABASE</td>
<td>NUMBER</td>
<td>Time deletion of Free Autonomous Database</td>
</tr>
</tbody>
</table>
OCI_BUDGET_ALERT_RULES View

OCI_BUDGET_ALERT_RULES describes all the Oracle Cloud Infrastructure budget alert rules in the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.

Queries against this view return results only if you have budgets and budget alerts created in your tenancy.

See Budgets Overview for more information.

To query this view you need a dynamic group that includes your Autonomous Database instance and the following policy defined on that dynamic group:

Allow dynamic-group <group-name> to read usage-budgets in tenancy

This policy lets you list budget summary and budget alerts in your tenancy (if you created a budget and a budget alert). Optionally you can restrict the result returned by querying the view to a given compartment:

Allow dynamic-group <group-name> to read usage-budgets in compartment <compartment-name>

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUDGETID</td>
<td>VARCHAR2</td>
<td>The OCID of the budget</td>
</tr>
<tr>
<td>REGION</td>
<td>VARCHAR2</td>
<td>Region name</td>
</tr>
<tr>
<td>COMPARTMENTID</td>
<td>VARCHAR2</td>
<td>The compartment ID in which the bucket is authorized</td>
</tr>
<tr>
<td>DEFINEDTAGS</td>
<td>CLOB</td>
<td>Defined tags for the resource</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>VARCHAR2</td>
<td>The description of the alert rule</td>
</tr>
<tr>
<td>DISPLAYNAME</td>
<td>VARCHAR2</td>
<td>The name of the alert rule</td>
</tr>
<tr>
<td>FREEFORMTAGS</td>
<td>CLOB</td>
<td>Free-form tags for the resource</td>
</tr>
</tbody>
</table>
OCI_BUDGET_SUMMARY View

OCI_BUDGET_SUMMARY describes all the Oracle Cloud Infrastructure budget summaries in the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.

Queries against this view return results only if you have budgets created in your tenancy.

See Budgets Overview for more information.

To query this view you need a dynamic group that includes your Autonomous Database instance and the following policy defined on that dynamic group:

Allow dynamic-group <group-name> to read usage-budgets in tenancy

This policy lets you list budget summary and budget alerts in your tenancy (if you created a budget and a budget alert). Optionally you can restrict the result returned by querying the view to a given compartment:

Allow dynamic-group <group-name> to read usage-budgets in compartment <compartment-name>

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGION</td>
<td>VARCHAR2</td>
<td>Region name</td>
</tr>
<tr>
<td>COMPARTMENTID</td>
<td>VARCHAR2</td>
<td>The OCID of the compartment</td>
</tr>
<tr>
<td>AMOUNT</td>
<td>NUMBER</td>
<td>The amount of the budget, expressed in the currency of a rate card</td>
</tr>
<tr>
<td>DEFINEDTAGS</td>
<td>CLOB</td>
<td>Defined tags for the resource</td>
</tr>
<tr>
<td>FREEFORMTAGS</td>
<td>CLOB</td>
<td>Free-form tags for the resource</td>
</tr>
<tr>
<td>DISPLAYNAME</td>
<td>CLOB</td>
<td>The display name of the budget</td>
</tr>
<tr>
<td>LIFECYCLESTATE</td>
<td>VARCHAR2</td>
<td>The current state of the budget</td>
</tr>
<tr>
<td>ACTUALSPEND</td>
<td>NUMBER</td>
<td>The actual spend in currency for the current budget cycle</td>
</tr>
<tr>
<td>ALERTRULECOUNT</td>
<td>NUMBER</td>
<td>The total number of alert rules in the budget</td>
</tr>
</tbody>
</table>
OCI_COST_DATA View

OCI_COST_DATA describes all the Oracle Cloud Infrastructure cost data for the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.

To query this view you need a dynamic group that includes your Autonomous Database instance and the following policy defined on that dynamic group:

Define tenancy usage-report as
ocid1.tenancy.oc1..aaaaaaaaned4fkpki6swj1r56u7cj63lf3wffbi6vqknstdvzub7vhqkggq
Endorse dynamic-group <group-name> to read objects in tenancy usage-report

Note:
Do not replace the OCID in this policy with another OCID. This usage-report OCID provides the Oracle Cloud Infrastructure usage data for your tenancy.
OCI_OBJECTSTORAGE_BUCKETS View

OCI_OBJECTSTORAGE_BUCKETS describes all the Oracle Cloud Infrastructure object storage buckets in the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.

To query this view you need a dynamic group that includes your Autonomous Database instance and the following policy defined on that dynamic group:

Allow dynamic-group <group-name> to read buckets in tenancy

This policy lets you list object storage buckets in your tenancy. Optionally you can restrict the result returned by querying this view to a given compartment:

Allow dynamic-group <group-name> to read buckets in compartment <compartment-name>

<table>
<thead>
<tr>
<th>Column</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGION</td>
<td>VARCHAR2</td>
<td>Region name</td>
</tr>
<tr>
<td>COMPARTMENTID</td>
<td>VARCHAR2</td>
<td>The compartment ID in which the bucket is authorized</td>
</tr>
<tr>
<td>Column</td>
<td>Datatype</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NAMESPACE</td>
<td>VARCHAR2</td>
<td>The Object Storage namespace in which the bucket resides</td>
</tr>
<tr>
<td>APPROXIMATECOUNT</td>
<td>NUMBER</td>
<td>The approximate number of objects in the bucket</td>
</tr>
<tr>
<td>APPROXIMATESIZE</td>
<td>NUMBER</td>
<td>The approximate total size in bytes of all objects in the bucket</td>
</tr>
<tr>
<td>AUTOTIERING</td>
<td>VARCHAR2</td>
<td>The auto tiering status on the bucket</td>
</tr>
<tr>
<td>CREATEDBY</td>
<td>VARCHAR2</td>
<td>The OCID of the user who created the bucket</td>
</tr>
<tr>
<td>DEFINEDTAGS</td>
<td>CLOB</td>
<td>Defined tags for the resource</td>
</tr>
<tr>
<td>FREEFORMTAGS</td>
<td>CLOB</td>
<td>Free-form tags for the resource</td>
</tr>
<tr>
<td>ETAG</td>
<td>VARCHAR2</td>
<td>The entity tag (ETag) for the bucket</td>
</tr>
<tr>
<td>ID</td>
<td>VARCHAR2</td>
<td>The OCID of the bucket</td>
</tr>
<tr>
<td>ISREADONLY</td>
<td>VARCHAR2</td>
<td>Whether or not this bucket is read only</td>
</tr>
<tr>
<td>KMSKEYID</td>
<td>VARCHAR2</td>
<td>The OCID of a master encryption key</td>
</tr>
<tr>
<td>METADATA</td>
<td>VARCHAR2</td>
<td>Arbitrary string keys and values for user-defined metadata</td>
</tr>
<tr>
<td>NAME</td>
<td>VARCHAR2</td>
<td>The name of the bucket</td>
</tr>
<tr>
<td>OBJECTEVENTSENABLED</td>
<td>VARCHAR2</td>
<td>Whether or not events are emitted for object state changes in this bucket</td>
</tr>
<tr>
<td>OBJECTLIFECYCLEPOLICY</td>
<td>VARCHAR2</td>
<td>The entity tag (ETag) for the live object lifecycle policy on the bucket</td>
</tr>
<tr>
<td>ETAG</td>
<td>VARCHAR2</td>
<td>The entity tag (ETag) for the bucket</td>
</tr>
<tr>
<td>PUBLICACCESSTYPE</td>
<td>VARCHAR2</td>
<td>The type of public access enabled on this bucket</td>
</tr>
<tr>
<td>REPLICATIONENABLED</td>
<td>VARCHAR2</td>
<td>Whether or not this bucket is a replication source</td>
</tr>
<tr>
<td>STORAGETIER</td>
<td>VARCHAR2</td>
<td>The storage tier type assigned to the bucket</td>
</tr>
<tr>
<td>TIMECREATED</td>
<td>VARCHAR2</td>
<td>The date and time the bucket was created</td>
</tr>
<tr>
<td>VERSIONING</td>
<td>VARCHAR2</td>
<td>The versioning status on the bucket</td>
</tr>
</tbody>
</table>

OCI_USAGE_DATA View

OCI_USAGE_DATA describes all the Oracle Cloud Infrastructure usage data for the Oracle Cloud Infrastructure tenancy obtained from the current Autonomous Database instance.

To query this view you need a dynamic group that includes your Autonomous Database instance and the following policy defined on that dynamic group:

Define tenancy usage-report as
ocid1.tenancy.oc1..aaaaaaaned4fkapkibwjl56u7cj631f3wffbelvqknstgvzub7vhhkggq
Endorse dynamic-group <group-name> to read objects in tenancy usage-report

**Note:**

Do not replace the OCID in this policy with another OCID. This usage-report OCID provides the Oracle Cloud Infrastructure cost and usage data for your tenancy.
## Autonomous Database – Oracle Database Features

Autonomous Database includes features that:

- Automate index management tasks, such as creating, rebuilding, and dropping indexes based on changes in the application workload.

  See [Manage Automatic Indexing on Autonomous Database](#) for more information.

**Note:**

There are restrictions for Automatic Indexing when you use JSON data with Autonomous Database. See [SODA Notes](#) for more information.

- Gather real-time statistics automatically while a conventional DML workload is running. Because statistics can go stale between stats gathering jobs, online
statistics gathering for conventional DML helps the optimizer generate more optimal plans. Online statistics aim to reduce the possibility of the optimizer being misled by stale statistics.

See Real-Time Statistics for more information.

• Gather statistics automatically on a more frequent basis. High-Frequency Automatic Optimizer Statistics Collection complements the standard statistics collection job. By default, the collection occurs every 15 minutes, meaning that statistics have less time in which to be stale. High-Frequency Automatic Optimizer Statistics Collection is enabled by default.

See Configuring High-Frequency Automatic Optimizer Statistics Collection for more information.

• Quarantine execution plans for SQL statements, for example, statements that are terminated by the Resource Manager for consuming excessive system resources in an Oracle Database. Automatic SQL Quarantine based on Resource Manager consumption limit violations is disabled by default but any manually quarantined SQL statement will be honored.

See Quarantine for Execution Plans for SQL Statements Consuming Excessive System Resources for more information.

• Automatically assess the opportunity for SQL plan changes to improve the performance for known statements.

See Managing the SPM Evolve Advisor Task for more information.

• Apache ORC format is supported in Autonomous Database for loading and querying data in object store.

See Create Credentials and Load Data Pump Dump Files into an Existing Table and Query External Data with ORC, Parquet, or Avro Source Files for more information.

• Complex types are supported in Autonomous Database for ORC, Avro, and Parquet structured files.

See DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types for more information.

Always Free Autonomous Database – Oracle Database 21c

Topics

• Always Free Autonomous Database Oracle Database 21c Features
• Always Free Autonomous Database Oracle Database 21c Notes

Always Free Autonomous Database Oracle Database 21c Features

When you provision Always Free Autonomous Database you can select either Oracle Database 19c or Oracle Database 21c.

Always Free Autonomous Database running with Oracle Database 21c offers many new innovative autonomous and developer-oriented functionality, including but not limited to the following:
Performance Features

• **Automatic Zone Maps**

Automatic zone maps are created and maintained for any user table without any customer intervention. Zone maps allow the pruning of block ranges and partitions based on the predicates in the queries. Automatic zone maps are maintained for direct loads, and are maintained and refreshed for any other DML operation incrementally and periodically in the background.

The feature is enabled as follows:

```
exec dbms_auto_zonemap.configure('AUTO_ZONEMAP_MODE','ON');
```

The feature is disabled as follows:

```
exec dbms_auto_zonemap.configure('AUTO_ZONEMAP_MODE','OFF');
```

See [Summary of DBMS_AUTO_ZONEMAP Subprograms](#) for more information.

• **Object Activity Tracking System**

Object Activity Tracking System (OATS) tracks the usage of various types of database objects. Usage includes operations such as access, data manipulation, or refresh.

No manual intervention is required to enable OATS, and zero or minimal configuration is required. See PL/SQL procedure DBMS_ACTIVITY.CONFIGURE and database dictionary views DBA_ACTIVITY_CONFIG for details.

Application Development: Advanced Analytical SQL Capabilities

• **SQL Macros**

SQL Macros, the capability to factor out common SQL constructs supports scalar expressions, increasing developer productivity, simplify collaborative code development, and improve code quality. See [SQL Macros](#) for more information.

• **Enhanced Analytic Functions**

Window functions support the full ANSI Standard, including the support of EXCLUDE options and the WINDOW clause. Supporting the full ANSI standard enables easier migration of applications that were developed with other standard-compliant database systems. See [Windowing Functions](#) for more information.

• **New Analytical and Statistical Aggregate Functions**

Several new analytical and statistical aggregate functions are available in SQL in Oracle Database 21c. With these additional SQL aggregation functions, you can write more efficient code and benefit from faster in-database processing.

- **CHECKSUM** computes the checksum of the input values or expression.
  - Supports the keywords ALL and DISTINCT.
- **KURTOSIS** functions KURTOSIS_POP and KURTOSIS_SAMP measure the tailedness of a data set where a higher value means more of the variance within the data set is the result of infrequent extreme deviations as opposed to frequent modestly sized deviations. Note that a normal distribution has a kurtosis of zero.
Supports the keywords ALL, DISTINCT, and UNIQUE.

- **SKEWNESS** functions **SKEWNESS_POP** and **SKEWNESS_SAMP** are measures of asymmetry in data. A positive skewness is means the data skews to the right of the center point. A negative skewness means the data skews to the left.

Supports the keywords ALL, DISTINCT, and UNIQUE.

- **ANY_VALUE**, a function to simplify and optimize the performance of **GROUP BY** statements, returns a random value in a group and is optimized to return the first value in the group. It ensures that there are no comparisons for any incoming row and eliminates the necessity to specify every column as part of the **GROUP BY** clause.


**Bitwise Aggregate Functions**

With the new bitwise type processing functions **BIT_AND_AGG**, **BIT_OR_AGG**, and **BIT_XOR_AGG**, native bitwise type processing is provided by Oracle Database 21c. These functions enable a type of processing inside the database for new types of application processing, improving the overall performance, avoiding unnecessary data movement, and natively taking advantage of core database functionality such as parallel processing. See Oracle Database 21c SQL Language Reference Guide for more information.

**JavaScript Execution using DBMS_MLE**

The **DBMS_MLE** package allows users to execute JavaScript code inside the Oracle Database and exchange data seamlessly between PL/SQL and JavaScript. The JavaScript code itself can execute PL/SQL and SQL through built-in JavaScript modules. JavaScript data types are automatically mapped to Oracle Database data types and vice versa.

With the **DBMS_MLE** package, developers can write their data processing logic in JavaScript. JavaScript is a widely-used and popular programming language that can now also be used for writing programs that need to execute close to the data.

See DBMS_MLE for more information.

**Blockchain Table**

Blockchain tables are append-only tables in which only insert operations are allowed. Deleting rows is either prohibited or restricted based on time. Rows in a blockchain table are made tamper-resistant by special sequencing and chaining algorithms. Users can verify that rows have not been tampered. A hash value that is part of the row metadata is used to chain and validate rows.

Blockchain tables enable you to implement a centralized ledger model where all participants in the blockchain network have access to the same tamper-resistant ledger.

A centralized ledger model reduces administrative overheads of setting up a decentralized ledger network, leads to a relatively lower latency compared to decentralized ledgers, enhances developer productivity, reduces the time to market, and leads to significant savings for the organization. Database users can continue to use the same tools and practices that they would use for other database application development.

See Managing Blockchain Tables for more information.

**JSON Document Store Enhancements**

**Enhancements to Data Guide**
Enhances development flexibility and allows for materialized views, which may improve query performance with a trade-off against DML performance.

- **JSON_DATAGUIDE now gathers statistic information** if you specify `DBMS_JSON.GATHER_STATS` in the third argument. They are computed dynamically (up-to-date) at the time of the function call.

- **DBMS_JSON.CREATE_VIEW now gives you the option** to create a materialized view instead of a standard view. It also gives you the option to specify a particular path so the view can be created on a subset of the data. Both `CREATE_VIEW` and `ADD_VIRTUAL_COLUMN` are enhanced to allow automatic resolution of column naming conflicts, to provide a prefix to be applied to column names, and to specify the case-sensitivity of column names.

See [JSON Data Guide](#) for more information.

**Multivalue Index for JSON Data Type**

A new create index syntax `CREATE MULTIVALUE INDEX` allows you to create a functional index on arrays of strings or numbers within a JSON datatype column. Each unique value within the array will become a searchable index entry. This avoids the need for full JSON scans to find values within arrays in JSON columns, when searched using the `JSON_EXISTS` or `JSON_VALUE` operators. It provides similar benefits to conventional functional indexes when searching JSON, but conventional functional indexes are limited to a single indexed value per row.

See [Creating Multivalue Function-Based Indexes for JSON_EXISTS](#) and [Using a Multivalue Function-Based Index](#) for more information.

**New JSON Data Type**

JSON is a new SQL and PL/SQL data type for JSON data. Using this type provides a substantial increase in query and update performance. JSON data type uses binary format OSON that is optimized for SQL/JSON query and DML processing. Using the binary format can yield database performance improvements for processing JSON data.

You can use JSON data type and its instances in most places where a SQL data type is allowed, including:

- As the column type for table or view DDL
- With SQL/JSON functions and conditions, and with PL/SQL procedures and functions
- In Oracle dot-notation query syntax
- For creation of functional and search indexes

Oracle Call Interface and Java Database Connectivity (JDBC) clients now provide APIs that can work directly with binary JSON datatype OSON format, significantly saving network costs and server CPU cycles. Going forward, Oracle recommends using JSON datatype to store and process JSON data.

The [Oracle Autonomous JSON Database](#) uses OSON format to store and process JSON data.

See [Creating a Table With a JSON Column](#) for more information.

**New Oracle SQL Function JSON_TRANSFORM**

You can use SQL function `JSON_TRANSFORM` to update parts of a JSON document. You specify which parts to modify, the modifications, and any new values.
JSON_TRANSFORM is optimized by doing partial updates at OSON format level to achieve better JSON datatype update performance.

JSON_TRANSFORM makes it easier for an application to modify a JSON document, without having to parse and rebuild it. In most cases, it also avoids a round-trip between the server and client for the whole document.

See Oracle SQL Function JSON_TRANSFORM for more information.

• SQL/JSON Syntax Improvements

You can now express more complex SQL/JSON queries and express some queries more succinctly:

– New SQL function JSON_SCALAR accepts a scalar instance of a SQL data type and returns a scalar JSON value as an instance of JSON data type.

– New JSON path-language item methods support JSON_SCALAR: float(), double(), binary(), ymInterval(), and dsInterval().

– The JSON path-language and dot-notation syntax support the aggregate item methods: avg(), count(), minNumber(), maxNumber(), minString(), maxString(), sum().

See Simple Dot-Notation Access to JSON Data and SQL/JSON Path Expression Item Methods for more information.

SODA Enhancements: New JSON Data Type

The default collection storage changes to the JSON data type. See Creating a Document Collection with SODA for PL/SQL for more information.

PL/SQL Enhancements

• PL/SQL is enhanced to help you program iteration controls using new iterators in loops and in qualified expressions.

  The new iterator constructs are clear, simple, understandable, and efficient.

  See PL/SQL Extended Iterators for more information.

Gradual Database Password Rollover for Applications

An application can change its database passwords without an administrator having to schedule downtime.

To accomplish this, a database administrator can associate a profile having a non-zero limit for the PASSWORD_ROLLOVER_TIME password profile parameter, with an application schema. This allows the database password of the application user to be altered while allowing the older password to remain valid for the time specified by the PASSWORD_ROLLOVER_TIME limit. During the rollover period of time, the application instance can use either the old password or the new password to connect to the database server. When the rollover time expires, only the new password is allowed.

In addition to the clause PASSWORD_ROLLOVER_TIME in the CREATE PROFILE and ALTER PROFILE statements, the ALTER USER statement has a clause, EXPIRE PASSWORD ROLLOVER PERIOD. The ACCOUNT_STATUS column of the DBA_USERS and USER_USERS data dictionary views have several statuses indicating values to indicate rollover status.

See Managing Gradual Database Password Rollover for Applications for more information.
Always Free Autonomous Database Oracle Database 21c Notes

If you are using Always Free Autonomous Database with Oracle Database 21c, the following Oracle Database 21c functionality is not currently supported:

• Automatic Materialized Views

Autonomous Database RMAN Recovery Catalog

You can use Oracle Autonomous Database as a Recovery Manager (RMAN) recovery catalog. A recovery catalog is a database schema that RMAN uses to store metadata about one or more Oracle databases.

Use Autonomous Database as an RMAN Recovery Catalog

Recovery Manager (RMAN) recovery catalog is preinstalled in Autonomous Database in schema RMAN$CATALOG. The preinstalled catalog version is based on the latest version of Oracle Database and is compatible with all supported Oracle database versions.

The recovery catalog contains metadata about RMAN operations for each registered target database. When RMAN is connected to a recovery catalog, RMAN obtains its metadata exclusively from the catalog.

Note:

Autonomous Database is not supported as an RMAN target database. An RMAN target database is an Oracle Database to which RMAN is connected with the TARGET keyword. A target database is a database on which RMAN is performing backup and recovery operations. See Backing Up and Restoring Autonomous Transaction Processing Cloud for information on Autonomous Database backup and recovery operations.

Access to RMAN Recovery Catalog

Access to the recovery catalog is provided through predefined user RMAN$VPC with the appropriate access to the recovery catalog only. The RMAN$VPC user is locked by default.

You can either proxy to the predefined user RMAN$VPC through the ADMIN user or explicitly unlock the preinstalled schema:

• ADMIN user proxy into RMAN$VPC using ADMIN user’s password:

  connect admin[rman$vpc]/password@connect_string
ADMIN user can set a password for RMAN$VPC. Then the RMAN$VPC user can directly connect:

connect admin/password@connect_string
alter user rman$vpc identified by password account unlock;
connect rman$vpc/password@connect_string

Use the RMAN Recovery Catalog

You can use the RMAN recovery catalog by connecting RMAN to the preinstalled recovery catalog. Registering a target database in the recovery catalog maintains the database’s records in the recovery catalog. For example, to register a target database:

RMAN> connect catalog rman$vpc/password@connect_string;
connected to recovery catalog database
recovery catalog schema version 21.01.00.00. is newer than RMAN version

RMAN> register database;
database registered in recovery catalog
starting full resync of recovery catalog

To use your Autonomous Database as a recovery catalog, it is recommended to connect with the LOW service.

See Registering a Database in the Recovery Catalog for more details about using the RMAN recovery catalog.

Notes for Users Migrating from Other Oracle Databases

Describes information that is useful when you are migrating from other Oracle Databases to Oracle Autonomous Database.

Topics

- Initialization Parameters
- SQL Commands
- Data Types
- PL/SQL Packages
- Oracle XML DB
- Oracle Text
- Oracle Flashback
- Oracle Database Real Application Security
- Oracle LogMiner
- Fast Ingest
- Choose a Character Set for Autonomous Database
Initialization Parameters

Autonomous Database configures database initialization parameters automatically when you provision a database. You do not need to set any initialization parameters to start using your service. But, you can modify some parameters if you need to.

List of Initialization Parameters that can be Modified

- APPROX_FOR_AGGREGATION
- APPROX_FOR_COUNT_DISTINCT
- APPROX_FOR_PERCENTILE
- AWR_PDB_AUTOFETCH_ENABLED
- CONTAINER_DATA
- CURRENT_SCHEMA (Allowed only with ALTER SESSION)
- CURSOR_SHARING
- DDL_LOCK_TIMEOUT
- FIXED_DATE
- IGNORE_SESSION_SET_PARAM_ERRORS
- LDAP_DIRECTORY_ACCESS
- MAX_IDLE_TIME
- MAX_STRING_SIZE (See Data Types for details)
- NLS.Calendar
- NLS.COMP
- NLS.CURRENCY
- NLS_DATE_FORMAT
- NLS_DATE_LANGUAGE
- NLS_ISO_CURRENCY
- NLS_LANGUAGE
- NLS_LENGTH_SEMANTICS
- NLS.NCHAR_CONV.EXCP
- NLS_NUMERIC_CHARACTERS
- NLS_SORT
- NLS_TERRITORY
- NLS_TIME_FORMAT
- NLS_TIME_TZ_FORMAT
- NLS_TIMESTAMP_FORMAT
- NLS_TIMESTAMP_TZ_FORMAT
- OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES (Allowed only with ALTER SESSION)
- OPTIMIZER_IGNORE_HINTS
- OPTIMIZER_IGNORE_PARALLEL_HINTS
- OPTIMIZER_MODE
- PLSCOPE_SETTINGS
- PLSQL_CCFLAGS
- PLSQL_DEBUG
- PLSQL_OPTIMIZE_LEVEL
- PLSQL_WARNINGS
- QUERY_REWRITE_INTEGRITY
- RESULT_CACHE_MODE
- ROUTE_OUTBOUND_CONNECTIONS
- SESSION.EXIT_ON_PACKAGE_STATE_ERROR
- SQL_TRACE (Allowed only with ALTER SESSION) See Perform SQL Tracing on Autonomous Database for details.
- STATISTICS_LEVEL (Allowed only with ALTER SESSION)
- SYSDATE_AT_DBTIMEZONE (Allowed only with ALTER SESSION)

For more information on initialization parameters see Oracle Database Reference. For more information on TIME_ZONE, see Oracle Database SQL Language Reference.
For more information on `OPTIMIZER_IGNORE_HINTS` and `OPTIMIZER_IGNORE_PARALLEL_HINTS`, see Manage Optimizer Statistics on Autonomous Database.

**SESSION_EXIT_ON_PACKAGE_STATE_ERROR**

`SESSION_EXIT_ON_PACKAGE_STATE_ERROR` enables or disables special handling for stateful PL/SQL packages running in a session.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter type</td>
<td>Boolean</td>
</tr>
<tr>
<td>Default Value</td>
<td>FALSE</td>
</tr>
<tr>
<td>Modifiable</td>
<td>ALTER SESSION</td>
</tr>
<tr>
<td>Range of values</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

`SESSION_EXIT_ON_PACKAGE_STATE_ERROR` specifies the handling for a stateful PL/SQL package running in a session. When such a package undergoes modification, such as during planned maintenance for Oracle-supplied objects, the sessions that have an active instantiation of the package receive the following error when they attempt to run the package:

ORA-4068: existing state of package has been discarded

However, the application code that receives the ORA-4068 error may not be equipped to handle this error with its retry logic.

Setting `SESSION_EXIT_ON_PACKAGE_STATE_ERROR` to TRUE provides different handling for this case. When `SESSION_EXIT_ON_PACKAGE_STATE_ERROR` is TRUE, instead of just raising the ORA-4068 error when the package state is discarded, the session immediately exits. This can be advantageous because many applications are able to handle session termination by automatically and transparently re-establishing the connection.

**SYSDATE_AT_DBTIMEZONE**

`SYSDATE_AT_DBTIMEZONE` enables special handling in a session for the date and time value returned in calls to `SYSDATE` and `SYSTIMESTAMP`. Depending on the value of `SYSDATE_AT_DBTIMEZONE`, you see either the date and time based on the default Autonomous Database time zone, Coordinated Universal Time (UTC), or based on the time zone that you set in your database.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter type</td>
<td>Boolean</td>
</tr>
<tr>
<td>Default Value</td>
<td>FALSE</td>
</tr>
<tr>
<td>Modifiable</td>
<td>ALTER SESSION</td>
</tr>
<tr>
<td>Range of values</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

Default Autonomous Database Time Zone

The default Autonomous Database time zone is Coordinated Universal Time (UTC) and by default calls to `SYSDATE` and `SYSTIMESTAMP` return the date and time in UTC.
In order to change database time zone, you can run the following statement. This example sets the database time zone to UTC-5.

```
ALTER DATABASE SET TIME_ZONE='-05:00';
```

**Note:**
You must restart the Autonomous Database instance for the change to take effect.

After you set the database time zone, by default `SYSDATE` and `SYSTIMESTAMP` continue to return date and time in UTC (`SYSDATE_AT_DBTIMEZONE` is `FALSE` by default). If you set `SYSDATE_AT_DBTIMEZONE` to `TRUE` in a session, `SYSDATE` and `SYSTIMESTAMP` return the database time zone.

See Setting the Database Time Zone for more information on using the `SET TIME_ZONE` clause with `ALTER DATABASE`.

**Using SYSDATE_AT_DBTIMEZONE in a Session**

When `SYSDATE_AT_DBTIMEZONE` is `FALSE` in a session, calls to `SYSDATE` and `SYSTIMESTAMP` return values based on the default Autonomous Database time zone, Coordinated Universal Time (UTC).

When `SYSDATE_AT_DBTIMEZONE` is `TRUE` in a session, calls to `SYSDATE` or `SYSTIMESTAMP` return the date and time based on the database time zone.

**Note:**
Setting `SYSDATE_AT_DBTIMEZONE` to `TRUE` only affects the use of `SYSDATE` and `SYSTIMESTAMP` as operators in application SQL (for example, in queries, DML, and CTAS operations).

**Example**

The following example returns dates and times for two different time zones, based on the `SYSDATE_AT_DBTIMEZONE` parameter value:

```
SQL> SELECT DBTIMEZONE FROM DUAL;

DBTIMEZONE
-05:00

SQL> ALTER SESSION SET SYSDATE_AT_DBTIMEZONE=FALSE;

Session altered.

SQL> SELECT SYSTIMESTAMP FROM DUAL;
```

```
27-JAN-22 06.59.45.708082000 PM GMT

SQL> ALTER SESSION SET SYSDATE_AT_DBTIMEZONE=TRUE;
Session altered.

SQL> SELECT SYSTIMESTAMP FROM DUAL;
SYSTIMESTAMP
27-JAN-22 02.14.47.578946000 PM -05:00

Note:

When a SYSDATE or SYSTIMESTAMP query is executed in SQL Worksheet of Database Actions, the time and date value that is returned is in UTC (when SYSDATE_AT_DBTIMEZONE parameter is set to TRUE or FALSE). To obtain the database time zone when working in Database Actions, use TO_CHAR() as follows:

SQL> SELECT TO_CHAR(SYSTIMESTAMP,'YYYY-MM-DD"T"HH24:MI:SS TZH":"TZM') FROM DUAL;

TO_CHAR(SYSTIMESTAMP,'YYYY-MM-DD"T"HH24:MI:SS TZH":"TZM')
2022-01-27T14:15:00 -05:00

SQL Commands

Autonomous Database allows most of the SQL commands available in Oracle Database. To ensure the security and the performance of Autonomous Database, some SQL commands are restricted.

This section provides a list of SQL command limitations that are required to protect security and for the performance integrity of Autonomous Databases. Most of the standard SQL and PL/SQL syntax and constructs available with Oracle Database work in Autonomous Databases.

Note:

If you try to use a restricted SQL command the system reports:

ORA-01031: insufficient privileges

This error indicates that you are not allowed to run the SQL command in Autonomous Database.
The following SQL statements are not available in Autonomous Database:

- **ADMINISTER KEY MANAGEMENT**: By default Autonomous Database uses Oracle-managed encryption keys. Using Oracle-managed keys, Autonomous Database creates and manages the encryption keys that protect your data and Oracle handles rotation of the TDE master key.

  If you want customer-managed keys, a master encryption key in the Oracle Cloud Infrastructure Vault is used to generate the TDE master key on Autonomous Database. See Managing Encryption Keys on Autonomous Database for more information.

- **CREATE TABLESPACE, ALTER TABLESPACE, and DROP TABLESPACE**: Autonomous Database automatically configures default data and temporary tablespaces for the database. Adding, removing, or modifying tablespaces is not allowed. Autonomous Database creates one tablespace or multiple tablespaces automatically depending on the storage size.

- **CREATE DATABASE LINK**

  Use `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to create database links in Autonomous Database. See Use Database Links with Autonomous Database for more information.

- **CREATE LIBRARY**

**SQL Statements with Restrictions in Autonomous Database**

The following DDL statements are available in Autonomous Database with some restrictions:

<table>
<thead>
<tr>
<th>SQL Command</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ALTER PLUGGABLE DATABASE</strong></td>
<td>Only the following clauses are allowed:</td>
</tr>
<tr>
<td>and <strong>ALTER DATABASE</strong></td>
<td>DATAFILE AUTOEXTEND ON</td>
</tr>
<tr>
<td></td>
<td>DATAFILE AUTOEXTEND OFF</td>
</tr>
<tr>
<td></td>
<td>DATAFILE RESIZE</td>
</tr>
<tr>
<td></td>
<td>DEFAULT EDITION</td>
</tr>
<tr>
<td></td>
<td>SET TIME_ZONE</td>
</tr>
<tr>
<td></td>
<td>SET CMU_WALLET</td>
</tr>
</tbody>
</table>

- **ALTER PROFILE**

  Using `ALTER PROFILE`, there are restrictions for a user defined `PASSWORD_VERIFY_FUNCTION`. See Manage Password Complexity on Autonomous Database for more information.

  Using `ALTER PROFILE`, the optional `CONTAINER` clause is ignored if specified.

  See Create Users on Autonomous Database for information on the password parameter values defined in the default profile.
<table>
<thead>
<tr>
<th>SQL Command</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTER SESSION</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 CONSTRAINTS</td>
</tr>
<tr>
<td></td>
<td>SET CURRENT_SCHEMA</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 OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES</td>
</tr>
<tr>
<td></td>
<td>SET ROW ARCHIVAL VISIBILITY</td>
</tr>
<tr>
<td></td>
<td>SET STATISTICS_LEVEL</td>
</tr>
<tr>
<td></td>
<td>SET TIME_ZONE</td>
</tr>
<tr>
<td>ALTER SYSTEM</td>
<td>ALTER SYSTEM is not allowed except ALTER SYSTEM SET and ALTER SYSTEM KILL SESSION</td>
</tr>
<tr>
<td></td>
<td>SET can only be used to set parameters listed in Initialization Parameters.</td>
</tr>
<tr>
<td>ALTER USER</td>
<td>The following clause is ignored: DEFAULT TABLESPACE</td>
</tr>
<tr>
<td></td>
<td>The IDENTIFIED clause cannot be used with the EXTERNALLY clause.</td>
</tr>
<tr>
<td></td>
<td>The IDENTIFIED BY VALUES clause is not allowed.</td>
</tr>
<tr>
<td>ALTER TABLE</td>
<td>For restrictions, see ALTER TABLE Restrictions.</td>
</tr>
<tr>
<td>CREATE PROFILE</td>
<td>PASSWORD_VERIFY_FUNCTION</td>
</tr>
<tr>
<td></td>
<td>See Manage Password Complexity on Autonomous Database for more information.</td>
</tr>
<tr>
<td></td>
<td>Using ALTER PROFILE, the optional CONTAINER clause is ignored if specified.</td>
</tr>
<tr>
<td></td>
<td>See Create Users on Autonomous Database for information on the password parameter values defined in the default profile.</td>
</tr>
<tr>
<td>CREATE TABLE</td>
<td>For restrictions, see CREATE TABLE Restrictions.</td>
</tr>
<tr>
<td>CREATE USER</td>
<td>The following clause is ignored:</td>
</tr>
<tr>
<td></td>
<td>•  DEFAULT TABLESPACE IDENTIFIED with the EXTERNALLY clause is not supported.</td>
</tr>
<tr>
<td></td>
<td>The IDENTIFIED BY VALUES clause is not allowed.</td>
</tr>
</tbody>
</table>

CREATE TABLE Restrictions

XMLType tables using XML schema-based storage are not allowed. See Oracle XML DB for more information.

The clauses not in this list are allowed.
<table>
<thead>
<tr>
<th>Clause</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster</td>
<td>Ignored</td>
</tr>
<tr>
<td>ilm_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>innmemory_table_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>LOB_storage_clause</td>
<td>The LOB_compression_clause is recognized. Other LOB_storage_clause parameters are ignored. See LOB_compression_clause for more information.</td>
</tr>
<tr>
<td>logging_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>organization external</td>
<td>Ignored</td>
</tr>
<tr>
<td>organization index</td>
<td>Creates a regular table with a primary key. Using the organization index clause does not create an index-organized table. You should test and verify the performance of the generated table for your application.</td>
</tr>
<tr>
<td>physical_properties</td>
<td>Ignored</td>
</tr>
</tbody>
</table>

**Note:**
For more information on CREATE TABLE, see *Database SQL Language Reference*.

### 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>allocate_extent_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>alter_iot_clauses</td>
<td>Ignored</td>
</tr>
<tr>
<td>deallocate_unused_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>ilm_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>innmemory_table_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>logging_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>modify_LOB_storage_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>physical_attributes_clause</td>
<td>Ignored</td>
</tr>
<tr>
<td>shrink_clause</td>
<td>Ignored</td>
</tr>
</tbody>
</table>

**Note:**
For more information on ALTER TABLE, see *Database SQL Language Reference*. 
Data Types

Autonomous Database allows most of the data types available in Oracle Database. To ensure the security and the performance of Autonomous Database, some data types are restricted.

The following data types are not supported or have limited support in Autonomous Database:

- Large Object (LOB) data types: only SecureFiles LOB storage is supported. BasicFiles LOBs are automatically converted to SecureFiles LOBs.
- Media types are not supported (Oracle Multimedia is desupported)

Checking and Setting MAX_STRING_SIZE

By default Autonomous Database uses extended data types and the value of MAX_STRING_SIZE is set to the value EXTENDED. With this setting you can specify a maximum size of 32767 bytes for the VARCHAR2, NVARCHAR2, and RAW data types. The default, EXTENDED, is the recommended setting and allows Autonomous Database to take full advantage of database capabilities.

Use DBMS_MAX_STRING_SIZE subprograms to check usage of extended data types and to change the database to revert to the older style STANDARD, supporting a maximum size of 4000 bytes for VARCHAR2, NVARCHAR2, and RAW data types.

Note:

Using DBMS_MAX_STRING_SIZE.MODIFY_MAX_STRING_SIZE is a one-way change that cannot be reverted. After a database is switched back to the STANDARD style of supporting a maximum length of 4000 bytes for the VARCHAR2, NVARCHAR2, and RAW data types, you cannot re-enable EXTENDED data types.

The ADMIN user is granted EXECUTE privilege with GRANT OPTION clause on DBMS_MAX_STRING_SIZE. Oracle recommends that you do not GRANT EXECUTE on this package to other users.

1. Check whether your environment can be reverted to the old style, STANDARD behavior:

   ```sql
   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:

   ```sql
   EXEC DBMS_MAX_STRING_SIZE.MODIFY_MAX_STRING_SIZE('STANDARD');
   
   See MODIFY_MAX_STRING_SIZE Procedure for more information.
   ```
See Extended Data Types for details on extended data types.

For a list of Oracle data types see Oracle Database SQL Language Reference.

### PL/SQL Packages

Notes for Oracle Database PL/SQL packages in Autonomous Database.

#### Unavailable PL/SQL Packages

- DBMS_DEBUG_JDWP
- DBMS_DEBUG_JDWP_CUSTOM
- UTL_INADDR

#### PL/SQL Packages Notes

- **DBMS_LDAP**
  - Specifying an IP address in the host name is not allowed.
  - The only allowed port is 636.
  - The SSLWRAP and SSLWALLETPASSWD arguments to the OPEN_SSL procedure are ignored. The default value for the SSLWRAP property is set to the wallet that is used by UTL_HTTP and DBMS_CLOUD for making outbound web requests on Autonomous Database.
  - DBMS_LDAP usage is audited by default. You cannot disable auditing for DBMS_LDAP.
  - The LDAP servers must be accessible from Autonomous Database through the public internet and the port 636 of the LDAP 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 LDAP servers through the internet.
  - SSL/TLS is enforced for all communication happening between LDAP server and Autonomous Database.
  - When your Autonomous Database instance is configured with a private endpoint, set the ROUTE_OUTBOUND_CONNECTIONS database parameter to 'PRIVATE_ENDPOINT' to specify that all outgoing LDAP connections are subject to the Autonomous Database instance private endpoint VCN's egress rules. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

- **UTL_HTTP**
  - 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 port is 443.

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_PROXY and SET_AUTHENTICATION_FROM_WALLET procedures are disallowed.

– The WALLET_PATH and WALLET_PASSWORD arguments for the CREATE_REQUEST_CONTEXT, REQUEST, and REQUEST_PIECES procedures are ignored.

– Oracle Wallet configuration cannot be altered. All arguments for SET_WALLET procedure are ignored.

– UTL_HTTP usage is audited by default. You cannot disable auditing for UTL_HTTP.

– When your Autonomous Database instance is configured with a private endpoint, set the ROUTE_OUTBOUND_CONNECTIONS database parameter to 'PRIVATE_ENDPOINT' to specify that all outgoing UTL_HTTP connections are subject to the Autonomous Database instance private endpoint VCN’s egress rules. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

• UTL_SMTP

  – The only supported email provider is Oracle Cloud Infrastructure Email Delivery service. See Overview of the Email Delivery Service for more information.

  – Mail with an IP address in the host name is not allowed.

  – The only allowed ports are 25 and 587.

  – UTL_SMTP usage is audited by default. You cannot disable auditing for UTL_SMTP.

  – When your Autonomous Database instance is configured with a private endpoint, set the ROUTE_OUTBOUND_CONNECTIONS database parameter to 'PRIVATE_ENDPOINT' to specify that all outgoing UTL_SMTP connections are subject to the Autonomous Database instance private endpoint VCN’s egress rules. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

• UTL_TCP

  – The IP address is not allowed in the host name.

  – The only allowed ports are: 443 (HTTP) 25 and 587 (SMTP).

  – For port 443, only HTTPS URLs are allowed.

  – The WALLET_PATH and WALLET_PASSWORD arguments for the OPEN_CONNECTION procedure are ignored. The default value for the WALLET_PATH and WALLET_PASSWORD property are set to the wallet that is used by UTL_HTTP and DBMS_CLOUD for making outbound web requests on Autonomous Database.

  – UTL_TCP usage is audited by default. You cannot disable auditing for UTL_TCP.

  – SSL/TLS is enforced for all communication happening over TCP/IP connections.

  – When your Autonomous Database instance is configured with a private endpoint, set the ROUTE_OUTBOUND_CONNECTIONS database parameter to ‘PRIVATE_ENDPOINT’ to specify that all outgoing UTL_TCP connections are subject to the Autonomous Database instance private endpoint VCN’s egress rules. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

• DBMS_NETWORK_ACL_ADMIN

  – Granting ACL privileges on IP addresses is not allowed.

  – The http_proxy and use_passwords ACL privileges are not allowed.

• 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>

See UTL_HTTP, DBMS_LDAP, UTL_SMTP, UTL_TCP, and DBMS_NETWORK_ACL_ADMIN in PL/SQL Packages and Types Reference for more information.

Oracle XML DB

Describes Autonomous Database support for Oracle XML DB features. To ensure the security and the performance of your Autonomous Database, some Oracle XML DB features are restricted.

The following is supported, in addition to the features listed:

- Full support for XMLQuery, XMLTable, and other SQL/XML standard functions
- Indexing schema including functional indexes using SQL/XML expressions, Structured XMLIndex and XQuery Full Text Index

---

**Note:**

If you migrate tables containing XMLType columns to Autonomous Database using Oracle Data Pump, you need to convert to Non-Schema Binary XML prior to using Oracle Data Pump Export (expdp).

---

<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>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>
</tbody>
</table>
For details on Oracle XML DB, see Oracle XML DB Developer's Guide.

### Oracle Text

Describes Autonomous Database support for Oracle Text features. To ensure the security and the performance of your Autonomous Database, some Oracle Text features are restricted.

<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 <code>ctx_report.query_log_summary</code></td>
<td>Not Supported</td>
<td>QUERY_LOG_SUMMARY</td>
</tr>
<tr>
<td>File Datastore</td>
<td>Not Supported (see replacement <code>DIRECTORY_DATASTORE</code>)</td>
<td>Datastore Types</td>
</tr>
<tr>
<td>URL Datastore</td>
<td>Not Supported (see replacement <code>NETWORK_DATASTORE</code>)</td>
<td>Datastore Types</td>
</tr>
<tr>
<td><strong>CREATE_INDEX</strong> with <strong>BIG_IO</strong> option</td>
<td>Supported if you grant the privilege to create a trigger to the user (<code>GRANT CREATE TRIGGER</code>),</td>
<td>Improved Response Time Using the <strong>BIG_IO</strong> Option of CONTEXT Index</td>
</tr>
<tr>
<td><strong>OPTIMIZE_INDEX</strong> in rebuild mode</td>
<td>Supported if you grant the privilege to create a trigger to the user (<code>GRANT CREATE TRIGGER</code>),</td>
<td><strong>OPTIMIZE_INDEX</strong></td>
</tr>
</tbody>
</table>

For details on Oracle Text, see Oracle Text Application Developer’s Guide.
Oracle Flashback

Oracle Flashback Technology is a group of Oracle Database features that let you view past states of database objects or to return database objects to a previous state without using point-in-time media recovery.

To restore and recover your database to a point in time, see Restore and Recover your Autonomous Database.

<table>
<thead>
<tr>
<th>Oracle Flashback Feature</th>
<th>Supported in Autonomous Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBMS_FLASHBACK</td>
<td>Yes except the procedure: DBMS_FLASHBACK.TRANSACTION_BACKOUT</td>
</tr>
<tr>
<td>Flashback Data Archive</td>
<td>No</td>
</tr>
<tr>
<td>Flashback Drop</td>
<td>Yes</td>
</tr>
<tr>
<td>Flashback Query</td>
<td>Yes</td>
</tr>
<tr>
<td>Flashback Table</td>
<td>Yes</td>
</tr>
<tr>
<td>Flashback Transaction</td>
<td>No</td>
</tr>
<tr>
<td>Flashback Transaction Query</td>
<td>Yes</td>
</tr>
<tr>
<td>Flashback Version Query</td>
<td>Yes</td>
</tr>
</tbody>
</table>

See About Oracle Flashback Technology for information on using Flashback features.

Oracle Database Real Application Security

Oracle Database Real Application Security is a database authorization model that: supports declarative security policies, enables end-to-end security for multitier applications, provides an integrated solution to secure database and application resources, and advances the security architecture of Oracle Database to meet existing and emerging demands of applications developed for the Internet.

See Introducing Oracle Database Real Application Security more information.

Real Application Security works the same on Autonomous Database as on an on-premises Oracle Database except you need to perform the following ADMIN tasks before using Real Application Security on Autonomous Database:

- To create Real Application Security users/roles, you need the PROVISION system privilege. As the ADMIN user run the following command to grant this privilege to a database user:

  ```sql
  SQL> EXEC
  XS_ADMIN_CLOUD_UTIL.GRANT_SYSTEM_PRIVILEGE('PROVISION','DB_USER');
  ```

  In this example, `DB_USER` is a database user.
Running this command on Autonomous Database replaces the following on-premise database command (note the _CLOUD_ is not in the following package name):

```sql
SQL> EXEC SYS.XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE('PROVISION', 'DB_USER', XS_ADMIN_UTIL.PTYPE_XS);
```

See General Procedures for Creating Application User Accounts for more information.

- To create Real Application Security data controls, you need the ADMIN_ANY_SEC_POLICY privilege. As the ADMIN user run the following command to grant this privilege:

```sql
EXEC
XS_ADMIN_CLOUD_UTIL.GRANT_SYSTEM_PRIVILEGE('ADMIN_ANY_SEC_POLICY', 'DB_USER');
```

In this example, _DB_USER_ is a database user.

Running this command on Autonomous Database replaces the following on-premise database command (note the _CLOUD_ is not in the following package name):

```sql
SQL> EXEC
SYS.XS_ADMIN_UTIL.GRANT_SYSTEM_PRIVILEGE('ADMIN_ANY_SEC_POLICY', 'DB_USER');
```

See Creating Roles and Application Users for more information.

**Oracle LogMiner**

Describes restrictions for Oracle LogMiner on Autonomous Database.

**Archived Log Retention Maximum is 48 Hours**

Autonomous Database archived log files are kept for up to 48 hours. LogMiner can only access up to 48 hours of archived log files.

If you attempt to mine log files older than 48 hours, LogMiner reports ORA-1285: "error reading file".

**Fast Ingest**

Fast ingest optimizes the processing of high-frequency, single-row data inserts into a database. Fast ingest uses the large pool for buffering the inserts before writing them to disk, so as to improve data insert performance.

For an overview of fast ingest and the steps involved in using this feature, refer to Using Fast Ingest in *Database Performance Tuning Guide*.

To use fast ingest with your database, you must enable the optimizer to use hints by setting the `optimizer_ignore_hints` parameter to `FALSE` at the session or system level, as appropriate.

Autonomous Database supports fast ingest, with the following limitations.

- Tables with the following characteristics can not use fast ingest:
– disk compression
– in-memory compression
– function-based indexes
– domain indexes
– bitmap indexes
– bitmap join indexes
– ref types
– varray types
– OID$ types
– unused columns
– LOBs that are stored out-of-line in a separate LOB segment
– Triggers
– binary columns
– foreign keys
– row archival
– invisible columns

• Following objects can not use fast ingest:
  – Temporary tables
  – Nested tables
  – Index organized tables
  – External tables
  – Materialized views with on-demand refresh

• The following partitioning types are not supported with fast ingest:
  – REFERENCE
  – SYSTEM

Choose a Character Set for Autonomous Database

The Autonomous Database default database character set is Unicode AL32UTF8 and the default national character set is AL16UTF16. When you provision a database, depending on the workload type, you can select a database character set and a national character set.

Note:

Oracle recommends using the default Unicode database character set (AL32UTF8) for its universality and compatibility with contemporary and future technologies and language requirements.

If you are using an on-premises database with a non-Unicode character set, migrating to the default Unicode character set can be a convoluted process requiring complex
data analysis. Thus, Autonomous Database lets you choose a character set when you provision an Autonomous Database instance.

1. Create an Autonomous Database following the provisioning steps. See Provision Autonomous Database for more information.

2. On the Create Autonomous Database page expand Show Advanced Options.

3. In the advanced options area, select the Management tab.

- In the Character Set field, use the selector to choose a character set.
- In the National Character Set field, use the selector to choose a national character set.

To make selecting a value easier, when you type in the text area this filters the list. For example, if you type JA you see only the options containing JA, including: JA16EUC, JA16EUCTILDE, JA16SJIS, JA16JA16SJISTILDE, and JA16VMS.

4. Click Create Autonomous Database to provision the Autonomous Database.

See the following for more information:

- Choosing an Oracle Database Character Set
- Character Set Migration
- Support Note 788156.1

Notes for Character Set Selection

Provides notes and limitations for selecting a character set and a national character set on Autonomous Database.

- When you provision an Autonomous Database instance you can only select a character set for Data Warehouse, Transaction Processing, or APEX workload types.

  The JSON Database workload type only supports the default character set and you cannot select a different character set.

- Always Free Autonomous Database does not support character set selection.

- You cannot select a different character set when you clone an instance. A cloned Autonomous Database instance has the same character set as the source database.
• You cannot change the character set of an existing Autonomous Database instance.

• APEX developer and administration pages are supported in English only when the database character set is different from the default (AL32UTF8).

Translation of user applications into other languages or setting an application’s primary language to a value other than English is not supported. If you use APEX in a language other than English, you may experience issues such as illegible, garbage text. You can, however, use your APEX applications to process non-English user data, as long as the languages of the data are supported by the selected database character set.

If you need full globalization support in APEX, migrate your applications and data to AL32UTF8, the universal Unicode character set, which is the default and recommended character set for Autonomous Database.

When you choose a character set other than the default Unicode character set (AL32UTF8) on Autonomous Database, the APEX language selector only shows languages that are supported in that database character set. For example, if you choose the database character set WE8ISO8859P1 (ISO 8859-1 West European), the language selector does not show Japanese or Korean.

• While you are provisioning an instance the Character Set selector on the Management tab lists the supported character set names. If you want to see a list of supported database character sets before you provision a database, refer to the following:
  – See Table A-4 in Recommended Database Character Sets
  – See Table A-6 in Other Character Sets

Database Features Unavailable in Autonomous Database

Lists the Oracle Database features that are not available in Autonomous Database. Additionally, database features designed for administration are not available.

List of Unavailable Oracle Features

• Oracle Real Application Security Administration Console (RASADM)
• Oracle OLAP: Not available in Autonomous Database. See Deprecation of Oracle OLAP for more information.
• Oracle R capabilities of Oracle Advanced Analytics
• Oracle Industry Data Models
• Oracle Database Lifecycle Management Pack
• Oracle Data Masking and Subsetting Pack
• Oracle Cloud Management Pack for Oracle Database
• Oracle Multimedia: Not available in Autonomous Database and deprecated in Oracle Database 18c.
• Oracle Sharding
• Oracle Workspace Manager
Migration is the process of copying the schema objects and data from a source MySQL or a third-party (non-Oracle) database, such as Amazon Redshift, Microsoft SQL Server, Sybase Adaptive Server, IBM DB2 (UDB), or Teradata, to Autonomous Database. Using Oracle SQL Developer Migration Wizard you can perform migration in an efficient and largely automated way.

Topics

- Migrating Amazon Redshift to Autonomous Database
- Migrating MySQL to Autonomous Database
- Migrating Microsoft SQL Server or Sybase Adaptive Server to Autonomous Database
- Migrating IBM DB2 (UDB) Database to Autonomous Database
- Migrating Teradata Database to Autonomous Database

Migrating Amazon Redshift to Autonomous Database

The SQL Developer Amazon Redshift Migration Assistant, available with SQL Developer 18.3 and later versions provides a framework for easy migration of Amazon Redshift environments on a per-schema basis.

This section describes the steps and the workflow for both an online migration of Amazon Redshift and for the generation of scripts for a scheduled, manual migration that you run at a later time.

Topics

- Autonomous Database Redshift Migration Overview
- Connect to Amazon Redshift
- Connect to Autonomous Database
- Start the Cloud Migration Wizard
- Review and Finish the Amazon Redshift Migration
- Use Generated Amazon Redshift Migration Scripts
- Perform Post Migration Tasks

Autonomous Database Redshift Migration Overview

Using SQL Developer you can migrate database files from Amazon Redshift to Autonomous Database.
• **Capture:** Captures Metadata schemas and tables from source database and stores in Migration Repository.

• **Convert:** Redshift Datatypes are mapped to Oracle Datatypes. Redshift Object names are converted to Oracle names based on Oracle Naming Convention. The Column Defaults that use Redshift functions are replaced by their Oracle equivalents.

• **Generate:** Generate schemas and DDLs based on the converted metadata.

• **Deploy:** Deploy the generated schemas and DDLs.

• **Copy Data:** Unload data from Redshift tables to Amazon Storage S3 then copy data from Amazon Storage to Autonomous Database tables (in schemas) that were Deployed earlier.

**Connect to Amazon Redshift**

Using SQL Developer you can migrate database Schemas and Tables from Amazon Redshift to Autonomous Database. To extract metadata and data from Amazon Redshift for a migration, you need to connect to Amazon Redshift with SQL Developer.

**Download Amazon Redshift JDBC Driver and Add the Third Party Driver**

1. Download an Amazon Redshift JDBC driver to access Amazon Redshift. Consult the Amazon Redshift documentation for the location of the most recent JDBC driver. For more information, see Configure a JDBC Connection.

   ![Note](Note.png)

   Use the Amazon Redshift JDBC Driver JDBC 4.2–compatible driver.

2. Store the Amazon Redshift JDBC driver in a local directory where SQL Developer can access the Amazon Redshift JDBC driver.

3. Add the Amazon Redshift JDBC driver as third party to SQL Developer before making a connection. Within SQL Developer, go to **Tools > Preferences > Database > Third Party JDBC Drivers** (for Mac, this is **Oracle SQL Developer > Preferences Database > Third Party JDBC Drivers**).

4. Click **Add Entry** and select the path to the Amazon Redshift JDBC Driver that you download.
5. Click **OK** to add the Amazon Redshift JDBC driver that you download.

Add Connection to Amazon Redshift Database

Connect to the Amazon Redshift database.

1. In the Connections panel, right-click **Connections** and select **New Connection**...
2. Select the Amazon Redshift tab and enter the connection information for Amazon Redshift.

If you are planning to migrate multiple schemas it is recommended to connect with the Master username to your Amazon Redshift system.

- For more details for configuring a JDBC Connection and obtaining the Amazon Redshift JDBC URL, see AWS: Configure a JDBC Connection.
- For more details for configuring security options for the connection (in case of "Amazon [500150] connection error"), see AWS: Configure Security options for Connection (in case of "Amazon [500150] connection error").
- If you deployed your Amazon Redshift environment within a Virtual Private Cloud (VPC) you have to ensure that your cluster is accessible from the Internet. See http://docs.aws.amazon.com/redshift/latest/gsg/rs-gsg-authorize-cluster-access.html for details of how to enable public Internet access.
- If your Amazon Redshift client connection to the database appears to hang or times out when running long queries, see http://docs.aws.amazon.com/redshift/latest/mgmt/connecting-firewall-guidance.html for possible solutions to address this issue.

Test the connection before you save it.

Additional Information for Amazon Authentication and Access Control

- AWS: Security
- AWS: Managing Cluster Security Groups Using the Console

**Connect to Autonomous Database**

Using SQL Developer you create a connection to Autonomous Database

Obtain the client credentials wallet zip file. For more information, see Download Client Credentials (Wallets).

1. In the Connections panel, right-click **Connections** and select **New Connection**...
2. Select the **Oracle** tab and enter the connection information for Autonomous Database.

3. For the AWS Redshift Migration connection, select the _low_ connection to your database.
   For more information, see Predefined Database Service Names for Autonomous Database.

4. Add a connection to Autonomous Database.

   ![New / Select Database Connection](image)

See Connect Oracle SQL Developer with a Wallet (mTLS) for more information.
Test the connection before you save it.

### Start the Cloud Migration Wizard

Invoke the Cloud Migration Wizard from the Tools menu of SQL Developer to initiate your Amazon Redshift migration to Autonomous Database.

The Cloud Migration wizard starts when you click Cloud Migrations from **Migration** in the **Tools** menu. The Cloud Migrations wizard enables you to migrate schemas, objects (tables), and data from an Amazon Redshift database to Autonomous Database.
The Cloud Migration Wizard is an easy set of steps. The Cloud Migration Wizard guides you to:

- Identify the schemas in your Amazon Redshift database that you want to migrate.
- Identify the target Autonomous Database.
- Define whether you want to migrate the metadata (DDL), the data, or both.
- Choose to migrate your system online, to generate scripts, or both.

**Identify the Amazon Redshift Database**

Identify the schemas in the Amazon Redshift database to migrate. All objects, mainly tables, in the schema will be migrated. Migration to Autonomous Database is on a per-schema basis. Schemas cannot be renamed as part of the migration.

1. In the AWS Redshift Migration, specify the Connection.
**Connection:** Name of the Redshift database connection.

**Available Schemas:** Schemas available for the specific connection.

**Selected Schemas:** Click the Add icon to select the schemas you want to migrate from Available Schemas.

**Include Data:** DDL and DATA migrates the selected schemas and data.

When you migrate data, you have to provide the AWS access key, AWS Secret Access Key, and an existing S3 bucket URI where the Redshift data will be unloaded and staged. The security credentials require certain privileges to store data in S3. It is recommended to create new, separate access keys for the migration. The same access key is used later on to load the data into the Autonomous Database using secure REST requests.

**AWS Access Key:** For more information on access keys, see AWS Identity and Access Management.

**AWS Secret Access:** For more information on access keys, see AWS Identity and Access Management.

**S3 Bucket URI:** For information on common S3 ServiceException errors, see S3ServiceException Errors.

For more information 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 Database See Connect to Autonomous Database. The user for this connection must have the administrative privileges; the connection is used throughout the migration to create schemas and objects.
The Amazon Redshift Migration Assistant allows you to do an online migration right away, to generate all scripts necessary for a migration, or both. If you chose to store the scripts in a local directory you have to specify the local directory (the directory must be writable by the user).

- **Connection**: Name of the Autonomous Data Warehouse Cloud connection. Create a connection for the Autonomous Database if required. The user must have administrative privileges since this connection is used throughout the migration to create schemas and objects.
- **Migration Repository Password**: Password for the migration repository that is installed in the Autonomous Database as part of the schema migration. Either use the pre-filled password or enter a new password.
- **Remove repository on successful migration**: Select this option to remove the repository after the migration is completed. The repository is not required after migration.
- **Migrate Now**: Select this option to perform an online migration immediately.
**Note:**

- If **Include Data** from Step 1 and **Migrate Now** are both unselected, you are opting for just generation of all required SQL Scripts for manual migration.
- If **Include Data** from Step 1 is unchecked and **Migrate Now** is selected, then all selected schemas and their tables will be deployed in Autonomous Database but data will not be loaded into tables.
- If **Include Data** from Step 1 and **Migrate Now** are both selected, then all selected schemas and their tables will be deployed in Autonomous Database and data will be loaded into tables.

- **Directory:** Specify the director to store the generated scripts necessary for the migration; this saves the scripts in a local directory.

**Advanced Settings (Optional)**

The default settings should work unless you want to control the format options when Unloading to S3 storage or Copying from S3 storage to Autonomous Database. For more information on Format Options, see [DBMS_CLOUD Package Format Options](#). To use advanced options, click **Advanced Settings**.

**Output Directory:** Enter the path or click Select Directory to select the directory or folder for the migration.
**Maximum Number of Threads:** Enter the number of parallel threads to enable when loading data to tables in Autonomous Database.

**Use Scheduler:** Select this option to enable the scheduler for migration. You can schedule jobs for data load migration operations from the AWS S3 bucket to Autonomous Database. You have the option to run the scheduled jobs immediately or at a future date and time. To monitor the data load scheduled jobs, use the Scheduler node in the Connections navigator.

**Migration Execution Choice:**
- **Immediate** runs the scheduler as soon as the Redshift migration is triggered.
- **Once** runs the scheduler on a future date. You specify the **Start Date** and **Time Zone**. By default, the Start Date displays the current date and time of the local system. To change the start date, use the calendar icon to double-click and select the date or use the spinner to highlight the date and then click the field to set it.

**Redshift Unload Options: Allow Overwrite:** If this option is enabled, the unload process will overwrite existing files, including the manifest file (lists the data files that are created by the unload process). By default, unload fails if there are files that can be overwritten.

**ADWC format options: Reject Limit:** Enter the number of rows to reject when loading data to tables in Autonomous Database. The migration operation will error out after the specified number of rows are rejected. The default is 0.

**Review and Finish the Amazon Redshift Migration**

The summary shows a summary of the information that you have specified.

To change any information, press **Back** as needed.
If you have chosen an immediate migration, then the dialog of the migration wizard stays open until the migration is finished. If you select generate scripts, the migration process generates the necessary scripts in the specified local directory, and does not run the scripts.

To perform the migration, click Finish

If the selected schema name in AWS Redshift already exists in Autonomous Database, the migration process excludes deploying these selected schemas and displays a dialog:

Summary: What The Migration Assistant Creates

- Creates a new Autonomous Database user using the `schema_name` from Redshift.
- Creates a new bucket on S3 based on the `schema name`. 
• Creates sub-folders on S3 for each table.

Use Generated Amazon Redshift Migration Scripts

When you choose to generate migration scripts a new subdirectory is created in the local directory specified in the migration Wizard. You can run these scripts in real time or use them for programmatic processing.

The directory contains the following scripts:

• redshift_s3unload.sql
• adwc_ddl.sql
• adwc_dataload.sql
• adwc_dataload_scheduler.sql

These scripts contain all necessary commands to migrate your Amazon Redshift system to Autonomous Database. You can run these scripts in real time or use them for programmatic processing.

Unload Your Amazon Redshift Data into S3

The first step of a successful migration is to unload your Amazon Redshift data into Amazon S3, which acts as a staging area. Script redshift_s3unload.sql has all the Amazon Redshift unload commands to unload the data using the access credentials and the S3 bucket that were specified in the Migration Wizard workflow.

Connect to your Amazon Redshift environment to run this script.

Create Your Data Warehouse Objects

To prepare your Autonomous Database create your empty data warehouse schema prior to loading data. The Amazon Redshift Migration Assistant converted all Amazon Redshift schema structures into Oracle structures in script adwc_ddl.sql.

The script must be executed while you are connected to your Autonomous Database as privileged user; for example, ADMIN.

By default, the schema created for the migration has the same name as the schema in Amazon Redshift. You must change the password to the valid password for the specified user either in the script or after the script runs. If you want to change the schema name then change the schema name and all references to the name.

Load Your Amazon Redshift Data into Your Oracle Autonomous Database

The script adwc_dataload.sql contains all the load commands necessary to load your unloaded Amazon Redshift data straight from S3 into your Autonomous Database.

Execute the script while connected to your Autonomous Database as a privileged user; for example ADMIN.

If you want to change the target schema name when you create your data warehouse objects then you must adjust the target schema names in this script accordingly.

Use of JOB SCHEDULER

SQL Developer provides a graphical interface for using the DBMS_SCHEDULER PL/SQL package to work with Oracle Scheduler objects. To use the SQL Developer scheduling

The Scheduler node for a connection appears in the Connections navigator and in the DBA navigator. Use ADWC ‘admin’ user to navigate which displays Scheduler objects owned by the ‘admin’ monitoring status of data load jobs.

Under ADWC ‘admin’ Connection → Scheduler → Jobs, you will see AWS Redshift to ADWC data load jobs are created with name <schema_name>_<table_name>.

To see the status of completion of each data load, please expand each scheduled job and check the status.

Also for more detailed information about data load operation see table MD_REPORT in SQLDEV_MIGREPOS schema that stores information about table columns: and

    OPERATION_ID, LOGFILE_TABLE, BADFILE_TABLE, SOURCE_SCHEMA_NAME,
    TARGET_SCHEMA_NAME, SOURCE_TABLE_NAME,

and

    TARGET_TABLE_NAME, SOURCE_TABLE_ROWS, TARGET_TABLE_ROWS_LOADED, ERROR
    MESSAGE,

and

    STATUS (COMPLETED or FAILED)

Redshift Migration Log and Report Files

After Redshift Migration, you will find three files:

- MigrationResults.log: Log file of Redshift migration
- readme.txt: file explains how to use the Generated Amazon Redshift Migration Scripts.
- redshift_migration_reportxxx.txt: Contains information about Migration, here is sample:

    OPERATION ID : 8566
    LOGFILE TABLE : COPY$8566_LOG
    BADFILE TABLE : COPY$8566_BAD
    SOURCE_SCHEMA : sample
    TARGET_SCHEMA : SAMPLE
    SOURCE_TABLE : listing
    TARGET_TABLE : LISTING
    SOURCE_TABLE_ROWS : 192497
    TABLE_ROWS_LOADED : 192497
    ERROR MESSAGE : null
    STATUS : COMPLETED
    START TIME : 2018-09-27 17:25:18.662075
Perform Post Migration Tasks

After successful migration of your Redshift environment you should consider the following post-migration tasks:

1. **Drop schema SQLDEV_MIGREPOS**
   
   As part of the schema migration the Migration Assistant installs a minimal migration repository in the target Autonomous Database. After the migration this account is no longer needed and can be dropped or alternatively locked.

2. **Drop the Amazon S3 bucket used for staging**
   
   Unless you desire to use the unloaded Redshift data otherwise you can drop the bucket containing the unloaded data.

3. **Harden the Amazon account used for accessing S3**
   
   You should inactivate the security access key used for S3 access unless needed for other purposes.

4. **Drop the database credential used for data loading from S3**
   
   The Amazon security credentials to access S3 are stored encrypted as database credential `REDSHIFT_DWCS_CREDS` in your Autonomous Database in the privileged user schema that was used for the migration. Oracle recommends you drop this credential after successful migration unless needed for other purposes. For more information, see `DROP_CREDENTIAL` Procedure.

5. **Harden your accounts in your Autonomous Database**
   
   For the new schema created as part of the migration with the Migration Assistant, ensure to change the passwords of these accounts or lock and expire them if they're solely used for data storage.

### Migrating MySQL to Autonomous Database

Migration is the process of copying the schema objects and data from a source MySQL to Autonomous Database. Using Oracle SQL Developer Migration Wizard you can perform migration in an efficient and largely automated way.

See SQL Developer: Migrating Third-Party Databases and Before Migrating From MySQL for more information.
Migrating Microsoft SQL Server or Sybase Adaptive Server to Autonomous Database

Migration is the process of copying the schema objects and data from a source Microsoft SQL Server or Sybase Adaptive Server to Autonomous Database. Using Oracle SQL Developer Migration Wizard you can perform migration in an efficient and largely automated way.

See SQL Developer: Migrating Third-Party Databases and Before Migrating From Microsoft SQL Server or Sybase Adaptive Server for more information.

Migrating IBM DB2 (UDB) Database to Autonomous Database

Migration is the process of copying the schema objects and data from a source IBM DB2 (UDB) database to Autonomous Database. Using Oracle SQL Developer Migration Wizard you can perform migration in an efficient and largely automated way.

See SQL Developer: Migrating Third-Party Databases and Before Migrating From IBM DB2 for more information.

Migrating Teradata Database to Autonomous Database

Migration is the process of copying the schema objects and data from a source Teradata database to Autonomous Database. Using Oracle SQL Developer Migration Wizard you can perform migration in an efficient and largely automated way.

See SQL Developer: Migrating Third-Party Databases and Before Migrating From Teradata for more information.
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;
```

```sql
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;
```

```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_weeknuminyear = 6
and d_year = 1994
and lo_discount between 5 and 7
and lo_quantity between 26 and 35;

select sum(lo_revenue), d_year, p_brand1
from ssb.lineorder, ssb.dwdate, ssb.part, ssb.supplier
where lo_orderdate = d_datekey
and lo_partkey = p_partkey
and lo_suppkey = s_suppkey
and p_category = 'MFGR#12'
and s_region = 'AMERICA'
group by d_year, p_brand1
order by d_year, p_brand1;

select sum(lo_revenue), d_year, p_brand1
from ssb.lineorder, ssb.dwdate, ssb.part, ssb.supplier
where lo_orderdate = d_datekey
and lo_partkey = p_partkey
and lo_suppkey = s_suppkey
and p_brand1 between 'MFGR#2221' and 'MFGR#2228'
and s_region = 'ASIA'
group by d_year, p_brand1
order by d_year, p_brand1;

select sum(lo_revenue), d_year, p_brand1
from ssb.lineorder, ssb.dwdate, ssb.part, ssb.supplier
where lo_orderdate = d_datekey
and lo_partkey = p_partkey
and lo_suppkey = s_suppkey
and p_brand1 = 'MFGR#2221'
and s_region = 'EUROPE'
group by d_year, p_brand1
order by d_year, p_brand1;

select c_nation, s_nation, d_year, sum(lo_revenue) as revenue
from ssb.customer, ssb.lineorder, ssb.supplier, ssb.dwdate
where lo_custkey = c_custkey
and lo_suppkey = s_suppkey
and lo_orderdate = d_datekey
and c_region = 'ASIA' and s_region = 'ASIA'
and d_year >= 1992 and d_year <= 1997
group by c_nation, s_nation, d_year
order by d_year asc, revenue desc;

select c_city, s_city, d_year, sum(lo_revenue) as revenue
from ssb.customer, ssb.lineorder, ssb.supplier, ssb.dwdate
where lo_custkey = c_custkey
and lo_suppkey = s_suppkey
and lo_orderdate = d_datekey
and c_nation = 'UNITED STATES'
and s_nation = 'UNITED STATES'
and d_year >= 1992 and d_year <= 1997
  group by c_city, s_city, d_year
  order by d_year asc, revenue desc;

select c_city, s_city, d_year, sum(lo_revenue) as revenue
from ssb.customer, ssb.lineorder, ssb.supplier, ssb.dwdate
where lo_custkey = c_custkey
  and lo_suppkey = s_suppkey
  and lo_orderdate = d_datekey
  and (c_city='UNITED KI1' or c_city='UNITED KI5')
  and (s_city='UNITED KI1' or s_city='UNITED KI5')
  and d_year >= 1992 and d_year <= 1997
  group by c_city, s_city, d_year
  order by d_year asc, revenue desc;

select 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_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_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.

```sql
SELECT
  dwdate_hier.member_name as year,
  part_hier.member_name as part,
  customer_hier.c_region,
  customer_hier.member_name as customer,
  lo_quantity,
  lo_revenue
FROM  ssb.ssb_av
HIERARCHIES
  (dwdate_hier,
   part_hier,
   customer_hier)
WHERE
  dwdate_hier.d_year = '1998'
AND dwdate_hier.level_name = 'MONTH'
AND part_hier.level_name = 'MANUFACTURER'
AND customer_hier.c_region = 'AMERICA'
AND customer_hier.level_name = 'NATION'
ORDER BY
  dwdate_hier.hier_order,
  part_hier.hier_order,
  customer_hier.hier_order;

SELECT
  dwdate_hier.member_name as time,
  part_hier.member_name as part,
  customer_hier.member_name as customer,
  supplier_hier.member_name as supplier,
  lo_quantity,
  lo_supplycost
FROM  ssb.ssb_av
HIERARCHIES
  (dwdate_hier,
   part_hier,
   customer_hier, supplier_hier)
WHERE
  (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;
```
WITH 
  h1 AS (SELECT 
    dwdate_hier.member_name as year, 
    part_hier.member_name as part, 
    supplier_hier.member_name as supplier, 
    lo_quantity, 
    lo_revenue, 
    lo_supplycost 
  FROM  ssb.ssb_av
  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, 
    supplier_hier.hier_order)

SELECT 
  h1.year, 
  h1.part, 
  h1.supplier, 
  h1.lo_quantity, 
  h1.lo_revenue, 
  h1.lo_supplycost
FROM h1
ORDER BY 
  h1.year, 
  h1.part, 
  h1.supplier;
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 database, the collection metadata includes the following information (regardless of which SODA implementation you use to create the default collection):

```
{
  "keyColumn" : {
    "name" : "ID",
    "sqlType" : "VARCHAR2",
    "maxLength" : 255,
    "assignmentMethod" : "UUID"
  },

  "contentColumn" : {
    "name" : "JSON_DOCUMENT",
    "sqlType" : "BLOB",
    "jsonFormat" : "OSON"
  },

  "versionColumn" : {
    "name" : "VERSION",
    "method" : "UUID"
  },

  "lastModifiedColumn" : {
    "name" : "LAST_MODIFIED"
  },

  "creationTimeColumn" :
```
Note:

Using Always Free Autonomous Database with Oracle Database 21c, the default metadata changes as follows.

```json
{
    "keyColumn": {
        "name": "ID",
        "sqlType": "VARCHAR2",
        "maxLength": 255,
        "assignmentMethod": "UUID"
    },

    "contentColumn": {
        "name": "JSON_DOCUMENT",
        "sqlType": "JSON"
    },

    "versionColumn": {
        "name": "VERSION",
        "method": "UUID"
    },

    "lastModifiedColumn": {
        "name": "LAST_MODIFIED"
    },

    "creationTimeColumn": {
        "name": "CREATED_ON"
    },

    "readOnly": false
}
```
SODA Customized Collection Metadata on Autonomous Database

Describes SODA collection custom metadata on Autonomous Database.

Each SODA implementation provides a way to customize the collection metadata during collection creation. For example, in SODA for Java, you can use the following command:

```java
OracleDocument metadata = db.createDocumentFromString("metadata_string");
OracleCollection col = db.admin().createCollection("myCustomColl", metadata);
```

In this example, for `metadata_string` you can use the default metadata as the starting point, and customize the following:

- **Change `keyColumn.assignmentMethod` to `CLIENT`:** Change the value of the `assignmentMethod` under `keyColumn` in the metadata to `CLIENT` (instead of `UUID`).
  
  **Valid values for `keyColumn.assignmentMethod` on Autonomous Database:**
  
  - **UUID** (default): Keys are generated by SODA, based on the UUID.
  - **CLIENT**: Keys are assigned by the client application.

- **Provide a `mediaTypeColumn` name value:** A media type column is needed if the collection is to be heterogeneous, that is, it can store documents other than JavaScript Object Notation (JSON). See Media Type Column Name for details.

The following example specifies client-assigned keys and a custom media type column. The `mediaTypeColumn` name is specified with the value `YOUR_MEDIA_TYPE_COLUMN_NAME`. Otherwise, the default settings are used.

```json
{
  "keyColumn" : {
    "name" : "ID",
    "sqlType" : "VARCHAR2",
    "maxLength" : 255,
    "assignmentMethod" : "CLIENT"
  },

  "contentColumn" : {
    "name" : "JSON_DOCUMENT",
    "sqlType" : "BLOB"
  },

  "versionColumn" : {
    "name" : "VERSION",
    "method" : "UUID"
  },

  "lastModifiedColumn" : {
```
"name" : "LAST_MODIFIED"
},

"creationTimeColumn" :
{
  "name" : "CREATED_ON"
},

"mediaTypeColumn" :
{
  "name" : "YOUR_MEDIA_TYPE_COLUMN_NAME"
},

"readOnly" : false
Obtain Tenancy Details

When you file a service request or when you need Autonomous Database tenancy details for other reasons you can obtain these details for your instance. Tenancy details for an instance are available on the Oracle Cloud Infrastructure console or you can obtain these details by querying the database.

If you need to file a service request use Oracle Cloud Support or contact your support representative and provide the following information:

- Database Name
- Region
- Tenancy OCID
- Database OCID
- Compartment OCID
- Outbound IP Address

If you are connected to the database you can obtain tenancy details by querying the CLOUD_IDENTITY column of the V$PDBS view.

For example, running the following:

```
SELECT cloud_identity FROM v$pdbs;
```

Shows this output:

```
CLOUD_IDENTITY
-----------------------------------------------
{
   "DATABASE_NAME" : "DBxxxxxxxxxxxxx",
   "REGION" : "us-phoenix-1",
   "TENANT_OCID" : "OCID1.TENANCY.REGION1..ID1",
   "DATABASE_OCID" : "OCID1.AUTONOMOUSDATABASE.OC1.IAD.ID2",
   "COMPARTMENT_OCID" : "ocid1.tenancy.region1..ID3"
   "OUTBOUND_IP_ADDRESS" : 
   [   "192.0.2.254"
   ]
}
```

If your Autonomous Database instance was created before the tenancy details feature was added and has not been restarted, then this query does not return tenancy details. In this case, as a one-time operation, restart your instance and run the query again. You can restart your instance from the Oracle Cloud Infrastructure console or using the restart API.
Database Name

You set the database name when you provision a database or when you rename a database.

The Oracle Cloud Infrastructure Console shows the database name on the Autonomous Database Details page under General Information in the Database Name field.

Region

The Oracle Cloud Infrastructure Console shows the region on the console, in the Regions area.

Tenancy OCID

The tenancy details page shows the tenancy OCID.

See Managing the Tenancy for information on accessing the Tenancy Details page.

See Resource Identifiers for information on Oracle Cloud Identifiers.

Database OCID

The Oracle Cloud Infrastructure Console shows the database OCID on the Autonomous Database Details page under General Information in the OCID field.

See Resource Identifiers for information on Oracle Cloud Identifiers.

Compartment OCID

See Managing Compartments for more information on compartments.

See Resource Identifiers for information on Oracle Cloud Identifiers.

Outbound IP Address

The OUTBOUND_IP_ADDRESS is the outbound IP address of your Autonomous Database instance. You can use the OUTBOUND_IP_ADDRESS when you create a database link to another Autonomous Database instance that uses ACLs to restrict access. In this case, you need to allow the specified outbound IP address to connect to the target Autonomous Database and then create the database link.

The OUTBOUND_IP_ADDRESS shows the outbound IP address in the following cases:

- When your Autonomous Database instance uses a Public Endpoint.
- When your Autonomous Database instance uses a Private Endpoint and the database property ROUTE_OUTBOUND_CONNECTIONS is set to '' (the default value).
When your Autonomous Database instance uses a Private Endpoint and you set the database property `ROUTE_OUTBOUND_CONNECTIONS` to 'PRIVATE_ENDPOINT', outbound connections go through the private endpoint. See Enhanced Security for Outbound Connections with Private Endpoints for more information.

See Create Database Links from Autonomous Database to Publicly Accessible Oracle Databases for information on creating database links.

See Configuring Network Access with Access Control Rules (ACLs) for information on configuring access control rules.