Oracle® Cloud

Using Oracle Autonomous Data Warehouse on Dedicated Exadata Infrastructure

F22040-05
February 2020
## Contents

**Preface**

<table>
<thead>
<tr>
<th>Audience</th>
<th>viii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation Accessibility</td>
<td>viii</td>
</tr>
<tr>
<td>Related Documents</td>
<td>viii</td>
</tr>
<tr>
<td>Conventions</td>
<td>ix</td>
</tr>
</tbody>
</table>

### 1 Getting Started with Autonomous Data Warehouse

<table>
<thead>
<tr>
<th>Getting Started with Autonomous Data Warehouse</th>
<th>1-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>About Autonomous Data Warehouse</td>
<td>1-1</td>
</tr>
<tr>
<td>Before You Begin with Autonomous Data Warehouse</td>
<td>1-2</td>
</tr>
<tr>
<td>Key Features of Autonomous Data Warehouse</td>
<td>1-2</td>
</tr>
<tr>
<td>Typical Workflow for Using Autonomous Data Warehouse</td>
<td>1-4</td>
</tr>
<tr>
<td>Build Reports and Dashboards with Analytics in Autonomous Data Warehouse</td>
<td>1-4</td>
</tr>
<tr>
<td>Security and Authentication in Autonomous Data Warehouse</td>
<td>1-4</td>
</tr>
</tbody>
</table>

### 2 Connecting to Autonomous Data Warehouse

<table>
<thead>
<tr>
<th>Connecting to Autonomous Data Warehouse</th>
<th>2-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>About Connecting to an Autonomous Data Warehouse Dedicated Database</td>
<td>2-1</td>
</tr>
<tr>
<td>Connect to Autonomous Data Warehouse Using an Application</td>
<td>2-2</td>
</tr>
<tr>
<td>Connect to Autonomous Data Warehouse Using a Client Application</td>
<td>2-2</td>
</tr>
<tr>
<td>Prepare for Oracle Call Interface (OCI), ODBC, and JDBC OCI Connections</td>
<td>2-3</td>
</tr>
<tr>
<td>Prepare for JDBC Thin Connections</td>
<td>2-4</td>
</tr>
<tr>
<td>Applications with Built-In Support for Wallet ZIP Files</td>
<td>2-4</td>
</tr>
<tr>
<td>Download Client Credentials</td>
<td>2-5</td>
</tr>
<tr>
<td>Connect to Autonomous Data Warehouse Using Oracle Database Tools</td>
<td>2-6</td>
</tr>
<tr>
<td>Connect to Oracle SQL Developer Web in the Database</td>
<td>2-7</td>
</tr>
<tr>
<td>Connect with Oracle SQL Developer Version 18.2 or Later</td>
<td>2-8</td>
</tr>
<tr>
<td>Connect with Oracle SQL Developer Versions Earlier Than 18.2</td>
<td>2-9</td>
</tr>
<tr>
<td>Connect with Oracle SQLcl</td>
<td>2-11</td>
</tr>
<tr>
<td>Connect with SQL*Plus</td>
<td>2-13</td>
</tr>
<tr>
<td>Oracle Call Interface (OCI) Connections and Wallets</td>
<td>2-13</td>
</tr>
<tr>
<td>Predefined Database Service Names for Autonomous Data Warehouse Dedicated Databases</td>
<td>2-14</td>
</tr>
</tbody>
</table>
Connect Applications to Autonomous Data Warehouse 2-15
  Connect with Microsoft .NET and Visual Studio 2-15
  Connect with JDBC Thin Driver and UCP 2-16
  Connect with Python, Node.js, and other Scripting Languages 2-16
  Connect with Oracle Cloud Infrastructure FastConnect 2-19

3 Loading Data into Autonomous Data Warehouse

About Loading Data 3-1
Load Data from Files in the Cloud 3-1
  Create Credentials, Create the Table, and Copy Data 3-2
  Monitor and Troubleshoot Data Loading 3-3
Load Data Using Oracle Data Pump 3-4
  Export Data from Your Existing Oracle Database 3-4
  Upload the Export Files to Cloud Object Storage 3-5
  Import Data Using Oracle Data Pump 3-6
Replicate Data Using Oracle GoldenGate 3-7
Load Data Using SQL*Loader 3-8

4 Querying External Data with Autonomous Data Warehouse

Query External Data 4-1
Validate External Data 4-3
View Logs for Data Validation 4-4

5 Using Analytics and Visualization

Working with Oracle Analytics Desktop in Autonomous Data Warehouse 5-1

6 Exporting Data to Other Oracle Databases

Use Data Pump to Create a Dump File Set on Autonomous Data Warehouse 6-1
Move Dump File Set from Autonomous Data Warehouse to Your Cloud Object Store 6-3
Download Dump Files, Run Data Pump Import, and Clean Up Object Store 6-4

7 Creating Applications with Oracle Application Express in Autonomous Database

About Oracle Application Express 7-1
Access Oracle Application Express Administration Services 7-2
Create Oracle Application Express Workspaces in Autonomous Data Warehouse 7-4
Access Oracle Application Express App Builder 7-5
Use Web Services with Oracle Application Express 7-6
Send Email from Oracle Application Express 7-7
Restrictions and Limitations for Oracle Application Express with Autonomous Data Warehouse 7-8

8 Developing RESTful Services in Autonomous Database

About Oracle REST Data Services in Autonomous Database 8-1
Develop Oracle REST Data Services with Autonomous Database 8-1
Develop SODA for REST with Autonomous Database 8-2

9 Creating and Managing Directories

Create a Directory 9-1
Drop a Directory 9-2
List the Contents of a Directory 9-3
Copy Files Between Cloud Object Storage and a Directory 9-3

10 Managing Autonomous Data Warehouse Dedicated Databases

Create an Autonomous Data Warehouse Dedicated Database 10-1
View Details of an Autonomous Data Warehouse Dedicated Database 10-3
Add CPU or Storage Resources to an Autonomous Data Warehouse Dedicated Database 10-3
Remove CPU or Storage Resources from an Autonomous Data Warehouse Dedicated Database 10-3
Stop an Autonomous Data Warehouse Dedicated Database 10-4
Start an Autonomous Data Warehouse Dedicated Database 10-4
Restart an Autonomous Data Warehouse Dedicated Database 10-5
Clone an Autonomous Data Warehouse Dedicated Database 10-5
Move an Autonomous Data Warehouse Dedicated Database to a Different Compartment 10-7
Terminate an Autonomous Data Warehouse Dedicated Database 10-8

11 Managing Database Users

Create Database Users 11-1
Remove Database Users 11-2
Manage the ADMIN Database User 11-2
  Change the ADMIN Database User Password 11-2
  Unlock the ADMIN Database User Account 11-3
Manage Database User Privileges 11-3

ORACLE
Use Oracle Database Vault to Manage Database User Privileges 11-4
Configure and Enable Oracle Database Vault 11-4

12 Managing and Monitoring Performance
Use SQL Developer Web to Manage and Monitor Databases 12-1
Use Enterprise Manager to Manage and Monitor Databases 12-1
Use Performance Hub to Monitor Databases 12-2
Use Events and Notifications to Watch Resource Activity 12-4
Use Work Requests to Monitor Long-Running Operations 12-5
Manage Database Service Consumer Groups 12-5
  - Manage CPU/IO Shares 12-5
  - Manage Runaway SQL Statements 12-6
Manage Optimizer Statistics 12-6
Manage Automatic Indexing 12-7

13 Backing Up and Restoring Autonomous Data Warehouse
About Backup and Recovery on Autonomous Data Warehouse 13-1
Restore and Recover Your Autonomous Data Warehouse Dedicated Database 13-1
Back Up Your Autonomous Data Warehouse Dedicated Database Manually 13-2

A Cloud Object Storage URI Formats

B Using Oracle Database Features in Autonomous Data Warehouse Dedicated Deployments
About the Autonomous Data Warehouse Dedicated Database B-1
Latest Autonomous Data Warehouse Database Features B-2
The ADMIN User and the SYS User B-3
Modifying Database Initialization Parameters B-4
Oracle Database Features That Are Not Supported B-5
Oracle Database Features with Limited Support B-5
  - Limitations on the Use of Oracle Spatial and Graph B-6
  - Limitations on the Use of Oracle Text B-8
  - Limitations on the Use of Oracle XML DB B-8
Limitations on the Use of SQL Commands B-9
Limitations on the Use of Certain Data Types B-12
Limitations on the Use of Manually Created External Tables B-12
<table>
<thead>
<tr>
<th>Procedure/Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPY_DATA Procedure</td>
<td>C-1</td>
</tr>
<tr>
<td>CREATE_EXTERNAL_TABLE Procedure</td>
<td>C-2</td>
</tr>
<tr>
<td>DELETE_ALL_OPERATIONS Procedure</td>
<td>C-4</td>
</tr>
<tr>
<td>DELETE_FILE Procedure</td>
<td>C-4</td>
</tr>
<tr>
<td>DELETE_OBJECT Procedure</td>
<td>C-5</td>
</tr>
<tr>
<td>GET_OBJECT Procedure</td>
<td>C-5</td>
</tr>
<tr>
<td>LIST_FILES Function</td>
<td>C-7</td>
</tr>
<tr>
<td>LIST_OBJECTS Function</td>
<td>C-7</td>
</tr>
<tr>
<td>PUT_OBJECT Procedure</td>
<td>C-8</td>
</tr>
<tr>
<td>VALIDATE_EXTERNAL_TABLE Procedure</td>
<td>C-9</td>
</tr>
<tr>
<td>Format Parameter</td>
<td>C-10</td>
</tr>
</tbody>
</table>
Preface

This document describes how to use Autonomous Data Warehouse dedicated deployments and provides references to related documentation.

Audience

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

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

Autonomous Data Warehouse is built using Oracle Database 19c. Many database concepts and features of this service are further documented here:

Oracle Database 19c

For additional information, see these Oracle resources:

• Getting Started with Oracle Cloud
• User’s Guide for Oracle Data Visualization Desktop
• Oracle Cloud Infrastructure Object Storage Documentation
• Oracle Data Integration Platform Cloud
• Oracle Cloud Infrastructure Object Storage
• 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>monospace</td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
Getting Started with Autonomous Data Warehouse

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

Topics

• About Autonomous Data Warehouse
• Before You Begin with Autonomous Data Warehouse
• Key Features of Autonomous Data Warehouse
• Typical Workflow for Using Autonomous Data Warehouse
• Build Reports and Dashboards with Analytics in Autonomous Data Warehouse
• Security and Authentication in Autonomous Data Warehouse

About Autonomous Data Warehouse

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

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

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

Additionally, Autonomous Data Warehouse does not require any tuning. Autonomous Data Warehouse is designed as a "load and go" service: you create a data warehouse, define tables, load data, and then run queries. When you use Autonomous Data Warehouse, no tuning is necessary. You do not need to consider any details about parallelism, partitioning, indexing, or compression. The service automatically configures the database for high-performance queries.

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

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

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

Autonomous Data Warehouse also includes the following:

- Oracle Application Express (APEX): a low-code development platform that enables you to build scalable, secure enterprise apps with world-class features.
- Oracle REST Data Services (ORDS): a Java Enterprise Edition based data service that makes it easy to develop modern REST interfaces for relational data and JSON Document Store.
- Oracle SQL Developer Web: a browser-based interface of Oracle SQL Developer.

You can use Autonomous Data Warehouse with 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.

Before You Begin with Autonomous Data Warehouse

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

See Getting Started with Oracle Cloud.

Before you create Autonomous Data Warehouse dedicated deployments:

- On Oracle Cloud, purchase a subscription. You cannot create Autonomous Data Warehouse dedicated deployments until you do so.
- (Optional) if you want to leverage an object store for data loading you need your object store credentials to use with Oracle Autonomous Data Warehouse, including a username and a password. For details on the required credentials, depending on the object store you want to use, see the following:
  - Oracle Cloud Infrastructure Object Storage, the username is your Oracle Cloud Infrastructure user name. The password is your auth token. See Working with Auth Tokens.
  - Oracle Cloud Infrastructure Object Storage Classic, the username is your Oracle Cloud Infrastructure Classic user name and the password is your Oracle Cloud Infrastructure Classic password.

Key Features of Autonomous Data Warehouse

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

Key Features

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

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

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

- **Autonomous Data Warehouse supports**:
  - Existing applications, running in the cloud or on-premises
  - Connectivity via SQL*Net, JDBC, ODBC
  - Third-party data-integration tools
  - Oracle cloud services: Oracle GoldenGate Cloud Service, Oracle Integration Service, and others

- **High-performance features from the latest Oracle Database**: For more information, see [Latest Autonomous Data Warehouse Database Features](#).

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

- **Oracle SQL**: Autonomous Data Warehouse is compatible with existing applications that support Oracle Database.

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

**SQL Developer Autonomous Data Warehouse Support**

Using Autonomous Data Warehouse with SQL Developer you can do the following:

- Connect to Autonomous Data Warehouse
- Create tables in Autonomous Data Warehouse
- Load data into Autonomous Data Warehouse
- Copy tables to Autonomous Data Warehouse
- Transfer a schema to Autonomous Data Warehouse

**Business Intelligence Tools Support**

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

- Oracle Analytics Desktop
Third-party Business Intelligence tools

Typical Workflow for Using Autonomous Data Warehouse

Describes the steps to start using Autonomous Data Warehouse.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a Autonomous Data Warehouse</td>
<td>Create a Autonomous Data Warehouse dedicated deployment.</td>
<td>Create a Dedicated Autonomous Database</td>
</tr>
<tr>
<td>Create database users</td>
<td>Create accounts for database users and assign them appropriate privileges. Assign the necessary Autonomous Data Warehouse roles.</td>
<td>Create Database Users and Manage Database User Privileges</td>
</tr>
<tr>
<td>Load data into the database</td>
<td>Use the available tools to load data into the database.</td>
<td>Loading Data into Autonomous Data Warehouse</td>
</tr>
<tr>
<td>Connect to your database using a database client</td>
<td>Download a database client. Obtain security credentials and connect to your database.</td>
<td>Connecting to Autonomous Data Warehouse</td>
</tr>
</tbody>
</table>

Build Reports and Dashboards with Analytics in Autonomous Data Warehouse

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

For more information see Working with Oracle Analytics Desktop in Autonomous Data Warehouse.

Security and Authentication in Autonomous Data Warehouse

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

Connections to Autonomous Data Warehouse are made via SQL*Net. TCP and TCP with Secure Sockets Layer (SSL) security protocols are supported. TCP with SSL uses certificate-based authentication and SSL security protocol. This ensures that there is no unauthorized access to Autonomous Data Warehouse and that communications between the client and server are fully encrypted and cannot be intercepted or altered.

Certificate based authentication uses an encrypted key stored in a wallet on both the client (where the application is running) and the server (where your database is running). The key on the client must match the key on the server to make a connection. A wallet contains a collection of files, including the key and other information needed to connect to your database. For more information on connections to Autonomous Data Warehouse see About Connecting to an Autonomous Data Warehouse Dedicated Database.
You do not need to do any manual configuration to encrypt your data and the connections to your database. These are implemented by Autonomous Data Warehouse.

Autonomous Data Warehouse uses strong password complexity rules for all users based on Oracle Cloud security standards. For more information on the password complexity rules see Create Database Users.
Connecting to Autonomous Data Warehouse

Describes methods to securely connect to Autonomous Data Warehouse.

Topics

• About Connecting to an Autonomous Data Warehouse Dedicated Database
• Connect to Autonomous Data Warehouse Using a Client Application
• Download Client Credentials
• Connect to Autonomous Data Warehouse Using Oracle Database Tools
• Oracle Call Interface (OCI) Connections and Wallets
• Predefined Database Service Names for Autonomous Data Warehouse Dedicated Databases
• Connect Applications to Autonomous Data Warehouse
• Connect with Oracle Cloud Infrastructure FastConnect

About Connecting to an Autonomous Data Warehouse Dedicated Database

Applications and tools connect to an Autonomous Data Warehouse dedicated database using Oracle Net Services (also known as SQL*Net). Oracle Net Services enables a network session from a client application to an Oracle Database server.

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

Oracle Net Services support a variety of connection types to the Autonomous Data Warehouse, including:

• Oracle Call Interface (OCI), which is used by many applications written in C language. Examples include Oracle utilities such as Oracle SQL*Plus, SQL*Loader, and Oracle Data Pump.
• ODBC drivers, which can be used by applications running on Microsoft Windows, are layered over Oracle Call Interface (OCI).
• JDBC OCI, which is used by Java language applications. JDBC OCI adds a layer over Oracle Call Interface for Java applications. The Oracle SQLcl command-line interface uses JDBC OCI.
• JDBC Thin Driver, also for Java applications, is a pure Java driver. Oracle SQL Developer supports JDBC Thin Driver connections.
Third-party products and custom applications may use any of these connection types.

**Secure Connections to Autonomous Data Warehouse**

The network path to an Autonomous Data Warehouse dedicated database is through a VCN (virtual cloud network) and subnet defined by the dedicated infrastructure hosting the database. Usually, the subnet is defined as private, meaning that there is no public Internet access to databases.

Autonomous Data Warehouse provides several pairs of database services to use when connecting to your dedicated database. In each pair, one of the pair provides a secure TCP (TCPS) connection using the TLS protocol, and the other provides a TCP connection. In all other respects, the two members of a pair are the same. To ensure security of data in transit, Oracle strongly recommends that you use a secure connection, even if the database is only available through a private subnet. If you are familiar with using an Oracle Database within your own data center, you may not have previously used these secure connections.

To provide the secure connection, certification authentication uses an encrypted key stored in a *wallet* on both the client (where the application is running) and the server (where your Autonomous Data Warehouse dedicated 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 database. All communications between the client and the server are encrypted.

---

**Connect to Autonomous Data Warehouse Using a Client Application**

Autonomous Data Warehouse is preconfigured to support Oracle Net Services through a TNS listener is installed and configured to use either TCPS connections (secured using client credentials) or TCP connections.

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

**Topics**

- About Connecting to Autonomous Data Warehouse Using a Client Application
- Prepare for Oracle Call Interface (OCI), ODBC, and JDBC OCI Connections
- Prepare for JDBC Thin Connections
- Applications with Built-In Support for Wallet ZIP Files

**About Connecting to Autonomous Data Warehouse Using a Client Application**

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

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

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

3. Within your application, set up the connection.

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

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

You prepare for Oracle Call Interface (OCI), ODBC, and JDBC OCI connections by downloading and installing Oracle Instant Client and then downloading the client credentials for your Autonomous Data Warehouse dedicated database and making them available to Oracle Instant Client.

1. Download and install the Oracle Instant Client basic package for your system's OS and architecture:
   - **Oracle Linux**:
     a. Enable the Instant Client channel:
        ```
        yum -y install oracle-release-el7
        ```
     b. Download and install the Instant Client 19.3 basic package:
        ```
        yum -y install oracle-instantclient19.3-basic
        ```
        (To see a list of all Instant Client packages, go to [http://yum.oracle.com/repo/OracleLinux/OL7/oracle/instantclient/x86_64/index.html](http://yum.oracle.com/repo/OracleLinux/OL7/oracle/instantclient/x86_64/index.html).)
   - **Other OSes and architectures**:
     a. Go to the Oracle Instant Client Downloads page and select the download for your system's OS and architecture.
     b. On the download page, accept the Oracle Technology Network License Agreement, download the latest version of the Basic Package, and then install it by following the instructions at the bottom of the download page.

2. Download the zip file containing client credentials for your database to a secure directory on your computer.

   This zip file is available for download from the database's Details page in the Oracle Cloud console. Download the credentials as follows.
   a. In your web browser, sign in to Oracle Cloud and navigate to the Details page for the Autonomous Data Warehouse dedicated database.
   b. Click **DB Connection**.
   c. On the **Database Connection** page click **Download**.
   d. In the **Download Wallet** dialog, enter a wallet password in the **Password** field and confirm the password in the **Confirm Password** field.

      The password must be at least 8 characters long and must include at least 1 letter and either 1 numeric character or 1 special character.
   e. Click **Download** to save the client credentials zip file to a secure directory.

3. After downloading the zip file, follow these steps:
a. Unzip the client credentials zip file.

b. Edit the sqlnet.ora file provided in the client credentials, replacing "?/network/admin" with the full path of the directory where you unzipped the client credentials; for example, change:

   (DIRECTORY="?/network/admin")

to:

   (DIRECTORY="/users/jdoe/adbcredentials")

c. Create the TNS_ADMIN environment variable, setting its value to the full path of the directory where you unzipped the client credentials.

Prepare for JDBC Thin Connections

You prepare for JDBC thin connections by downloading and installing Oracle Database JDBC Driver and then downloading the client credentials for your Autonomous Data Warehouse dedicated database and making them available to Oracle Database JDBC Driver.

1. Go to the Oracle JDBC Downloads page and select the latest version of the drivers.

2. On the Driver & UCP Downloads page, accept the OTN License Agreement, and then download ojdbc10-full.tar.gz or ojdbc8-full.tar.gz, depending on the version of JDK you are using.

3. Download the zip file containing client credentials for your database to a secure directory on your computer.

   This zip file is available for download from the database's Details page in the Oracle Cloud console. Download the credentials as follows.

   a. In your web browser, sign in to Oracle Cloud and navigate to the Details page for the Autonomous Data Warehouse dedicated database.

   b. Click DB Connection.

   c. On the Database Connection page click Download.

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

      The password must be at least 8 characters long and must include at least 1 letter and either 1 numeric character or 1 special character.

   e. Click Download to save the client credentials zip file to a secure directory.

4. Unzip the client credentials zip file.

5. Create the TNS_ADMIN environment variable, setting its value to the full path of the directory where you unzipped the client credentials.

Applications with Built-In Support for Wallet ZIP Files

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

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

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

### Download Client Credentials

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

To download client credentials, do the following from Oracle Cloud Infrastructure console:

1. Navigate to the Autonomous Data Warehouse details page.
2. Click DB Connection.
3. On the Database Connection page click Download.
4. In the Download Wallet dialog, enter a wallet password in the Password field and confirm the password in the Confirm Password field.

The password must be at least 8 characters long and must include at least 1 letter and either 1 numeric character or 1 special character.
Note:

This password protects the downloaded Client Credentials wallet. This wallet is not the same as the Transparent Data Encryption (TDE) wallet for the database; therefore, use a different password to protect the Client Credentials wallet.

5. Click Download to save the client security credentials zip file.

By default the filename is: Wallet_<databasesname>.zip. You can save this file as any filename you want.

You must protect this file to prevent unauthorized database access.

The zip file includes the following:

- `tnsnames.ora` and `sqlnet.ora`: Network configuration files storing connect descriptors and SQL*Net client side configuration.
- `cwallet.ora` and `ewallet.p12`: Auto-open SSO wallet and PKCS12 file. PKCS12 file is protected by the wallet password provided in the UI.
- `keystore.jks` and `truststore.jks`: Java keystore and truststore files. They are protected by the wallet password provided while downloading the wallet.
- `ojdbc.properties`: Contains the wallet related connection property required for JDBC connection. This should be in the same path as `tnsnames.ora`.

Note:

Wallet files, along with the Database user ID and password provide access to data in your Autonomous Data Warehouse database. Store wallet files in a secure location. Share wallet files only with authorized users. If wallet files are transmitted in a way that might be accessed by unauthorized users (for example, over public email), transmit the wallet password separately and securely.

Connect to Autonomous Data Warehouse Using Oracle Database Tools

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

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

Topics

- Connect to Oracle SQL Developer Web in the Database
- Connect with Oracle SQL Developer Version 18.2 or Later
- Connect with Oracle SQL Developer Versions Earlier Than 18.2
- Connect with Oracle SQLcl
Connect to Oracle SQL Developer Web in the Database

Oracle SQL Developer Web, a browser-based application that provides many database development, management and monitoring features, is built into Autonomous Data Warehouse dedicated databases. Therefore, you can use it without downloading or installing additional software on your system.

By default, only the ADMIN database user has access to SQL Developer Web. Before you can sign in as another database user, the ADMIN user must enable access for that user, as described in Enabling User Access to SQL Developer Web in Using Oracle SQL Developer Web.

Note:

To sign into SQL Developer Web, the system you are using must have network access to the Autonomous Data Warehouse dedicated database.

To access the SQL Developer Web Sign In page, you copy its URL from the Database Connection dialog for the database and then paste the URL into your web browser's address bar.

1. In your web browser, sign in to Oracle Cloud and navigate to the Details page for the Autonomous Data Warehouse dedicated database.
2. Click DB Connection.
3. In the Database Connection dialog, click Application Connection.
4. In the Oracle SQL Developer Web box, click the Copy link in the Access URL field to copy the URL to your clipboard.
5. Paste the copied URL into your web browser's address bar.

For information about the wide-ranging features of SQL Developer Web, see Using Oracle SQL Developer Web.
Connect with Oracle SQL Developer Version 18.2 or Later

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

**Note:**
To connect Oracle SQL Developer to an Autonomous Data Warehouse dedicated database, the system running Oracle SQL Developer must have network access to the database.

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

1. Download the zip file containing client credentials for your database to a secure directory on system.
   
   This zip file is available for download from the database's Details page in the Oracle Cloud console. For more information, see [Download Client Credentials](#).

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

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

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

Connect with Oracle SQL Developer Versions Earlier Than 18.2

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

**Note:**

To connect Oracle SQL Developer to an Autonomous Data Warehouse dedicated database, the system running Oracle SQL Developer must have network access to the database.

Oracle recommends that you use Oracle SQL Developer version 18.2 or later; however, earlier versions of SQL Developer will work with Autonomous Data Warehouse.
To create a new connection to an Autonomous Data Warehouse dedicated database, do the following:

1. Download the zip file containing client credentials for your database to a secure directory on system.
   This zip file is available for download from the database's Details page in the Oracle Cloud console. For more information, see Download Client Credentials.
2. Start Oracle SQL Developer and in the connections panel, right-click Connections and select New Connection.

3. Choose the Connection Type Cloud PDB.
4. 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 PDB**.
- **Configuration File**: Click **Browse**, and select the client credentials zip file.
- **Keystore Password**: Enter the password generated while downloading the client credentials from Autonomous Data Warehouse.
- **Service**: Enter the service name. The client credentials file provides the service names.

**Connect with Oracle SQLcl**

SQLcl (Oracle SQL Developer Command Line) is a command-line interface for Oracle Database. It allows you to interactively or batch execute SQL and PL/SQL. SQLcl provides in-line editing, statement completion, and command recall for a feature-rich experience, all while also supporting your previously written SQL*Plus scripts.

**Note:**

To connect Oracle SQLcl to an Autonomous Data Warehouse dedicated database, the system running Oracle SQLcl must have network access to the database.

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

SQLcl can connect to an Autonomous Data Warehouse dedicated database using either an Oracle Call Interface (OCI) or a JDBC thin connection.

- If you use Oracle Call Interface (OCI), start by following the instructions in [Prepare for Oracle Call Interface (OCI), ODBC, and JDBC OCI Connections](#).
- If you use JDBC Thin, start by following the instructions in [Prepare for JDBC Thin Connections](#).

**SQLcl with Oracle Call Interface**

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

```
sql -oci
```

SQLcl: Release 18.4 Production on Wed Apr 03 15:28:40 2019

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

Username? (''?) sales_trans@atpc1_low
Password? (**********?) **************
Last Successful login time: Wed Apr 03 2019 15:29:19 -07:00

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

SQL>

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

**SQLcl with a JDBC Thin Connection**

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

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

   sql /nolog

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

   SQL> set cloudconfig directory/client_credentials.zip
   Wallet Password: **********

3. Connect to the Autonomous Data Warehouse database:

   SQL> connect username@servicename
   password

   For example:

   sql /nolog

   SQLcl: Release 18.4 Production on Thu Jan 24 11:29:41 2019

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

   SQL> set cloudconfig /home/atpc/wallet_ATPC1.zip
   Wallet Password: **********

   SQL> connect admin@atpc_medium

For more information, on the connection types specified in `tnsnames.ora`, see `Predefined Database Service Names for Autonomous Data Warehouse Dedicated Databases`.

For information on SQLcl, see *Oracle SQLcl*. 
Connect with SQL*Plus

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

Note:
To connect SQL*Plus to an Autonomous Data Warehouse dedicated database, the system running SQL*Plus must have network access to the database.

To install and configure the client and connect to an Autonomous Data Warehouse dedicated database using SQL*Plus, do the following:

1. Prepare to connect by following the instructions in Prepare for Oracle Call Interface (OCI), ODBC, and JDBC OCI Connections.
2. Connect using a database user, *password*, and database service name provided in the *tnsnames.ora* file.

For example:
```
sqlplus sales_trans@atpc1_low
```

Note:
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.

Oracle Call Interface (OCI) Connections and Wallets

Oracle Net Services can find the location of the Autonomous Data Warehouse wallet using the *WALLET_LOCATION* parameter in the *sqlnet.ora* file.

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

See Download Client Credentials for information on downloading client credentials for Autonomous Data Warehouse.

Predefined Database Service Names for Autonomous Data Warehouse Dedicated Databases

Autonomous Data Warehouse provides three pairs of database services to use when connecting to your dedicated database. In each pair, one of the pair provides a secure TCP (TCPS) connection using the TLS protocol, and the other provides a TCP connection. In all other respects, the two members of a pair are the same.

These three pairs of connection services are designed to support different kinds of database operations:

- **high_tls** and **high**: For high priority reporting and batch operations.
- **medium_tls** and **medium**: For typical reporting and batch operations.
- **low_tls** and **low**: For low priority reporting and batch operations.

**Note:**

Oracle strongly recommends that you use the secure TCP (TCPS) databases services. The TCP database services are provided for backward compatibility with applications that cannot use TLS-encrypted TCPS connections.

The following table compares these characteristics of these three pairs of database services:

- **Degree of Parallelism:** Parallel execution is a feature of Oracle Autonomous Database that creates and uses multiple processes to execute a single SQL statement. The **degree of parallelism** specifies the number of processes that can be used. Due to the performance cost of creating additional processes, parallel execution is generally worthwhile when:
  - The query references a large data set.
  - There is low concurrency.
  - Elapsed time is important.

- **Resource Shares:** Oracle Autonomous Database uses Oracle Database Resource Manager consumer groups to provide different resource allocations to each pair of database services. **Resources shares** define the guaranteed fraction of resources a consumer group will be granted. However, any consumer group can use more than the guaranteed amount of a resource when no resource contention exists.

- **Concurrent Statements:** The maximum number of SQL statements that the database can be processing for a consumer group before statements are queued.

- **FAN Enabled:** Whether Fast Application Notification is enabled by default.
• **TAC Enabled**: Whether Transparent Application Continuity is enabled by default.

<table>
<thead>
<tr>
<th>Service Names</th>
<th>Degree of Parallelism</th>
<th>Resource Shares</th>
<th>Concurrent Statements</th>
<th>FAN Enabled</th>
<th>TAC Enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>high_tls and high</td>
<td>CPU_COUNT</td>
<td>4</td>
<td>3</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>medium_tls and medium</td>
<td>4</td>
<td>2</td>
<td>1.25 × CPU_COUNT</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>low_tls and low</td>
<td>1</td>
<td>1</td>
<td>100 × CPU_COUNT</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

*CPU_COUNT* is the value of the *CPU_COUNT* initialization parameter for your dedicated database, which is equal to the **CPU Core Count** shown in the Oracle Cloud console.

---

**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 Data Warehouse performs more actions to configure the connection than just setting its consumer group.

---

**Connect Applications to Autonomous Data Warehouse**

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

**Topics**

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

**Connect with Microsoft .NET and Visual Studio**

Oracle Autonomous Data Warehouse supports connectivity to the Microsoft .NET Framework and Visual Studio.

Oracle Data Provider for .NET, Unmanaged Driver and Oracle Data Provider for .NET, Managed Driver provide run-time ADO.NET data access to the database service. Oracle Developer Tools for Visual Studio support design-time support for the database service. Oracle Providers for ASP.NET can store web application state using the service.

These software components are bundled together in a single software installation called Oracle Data Access Components (ODAC) for Windows and are available as a free download.

Oracle recommends using the latest ODAC version with Autonomous Data Warehouse.

**Set-up Instructions**
Click the following link for instructions on how to download, install, and configure ODAC for use with Autonomous Data Warehouse: Developing .NET Applications for Oracle Autonomous Database.

Connect with JDBC Thin Driver and UCP

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

See Prepare for JDBC Thin Connections for more information.

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

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

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

Install the Language Driver and Client Libraries

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

1. Install Instant Client or the Full Client. The minimum version supported for the Full Client is Version 11.2.0.4; for the Oracle Instant Client use version 12.1.0.2 or higher:

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

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

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

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

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

   d. On Windows, add the path to the PATH variable in the "System variables" section of the Environment Variables pane (for example add C: \instantclient_12_2). On Windows 8 access the PATH variable setting area by navigating to Control Panel>System>Advanced System Settings>Environment Variables. If you have multiple versions of Oracle libraries installed make sure the new directory occurs first in the path.
e. On non-Windows platforms, create a symbolic link if it does not exist. For example:

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

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

```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:$LD_LIBRARY_PATH
```

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

2. Install the relevant language driver for Oracle Database:

   - **Python**: To install cx_Oracle for Python, use the instructions on the following page: [cx_Oracle Installation](#).
   - **Node.js**: To install node-oracledb for Node.js, use the instructions on the following page: [Installing node-oracledb](#).
   - **ROracle**: To install ROracle for R, use the instructions on the following page: [ROracle](#).
   - **PHP**: To install PHP OCI8 for PHP, use the instructions on the following page: [Configuring PHP with OCI8](#).
     Windows DLLs are available on [http://php.net/downloads.php](http://php.net/downloads.php) and are also available from PECL oci8.
   - **PHP PDO_OCI**: To install PHP PDO_OCI for PHP, use the instructions on the following page: [Oracle Functions (PDO_OCI)](#).
     Windows DLLs are available on [http://php.net/downloads.php](http://php.net/downloads.php) included in PHP.
   - **Ruby**: To install ruby-oci8 for Ruby, use the instructions on the following page: [Install for Oracle Instant Client](#).
   - **DBD for Perl**: To install DBD::Oracle for Perl, set ORACLE_HOME and your library search path such as LD_LIBRARY_PATH or PATH to the Instant Client directory and use the instructions on the following page: [Installing DBD-Oracle](#).
Enable Oracle Network Connectivity and Obtain the Security Credentials (Oracle Wallet)

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

2. Extract the client credentials (wallet) files:
   a. Unzip the client credentials zip file.
   b. If you are using Instant Client, make a network/admin subdirectory hierarchy under the Instant Client directory if necessary. Then move the files to this subdirectory. For example depending on the architecture or your client system and where you installed Instant Client, the files should be in the directory:
      
      C:\instantclient_12_2\network\admin

      or

      /home/myuser/instantclient_18_5/network/admin

      or

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

   c. 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.

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, atpc_user, password, and atpc_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 with Oracle Cloud Infrastructure FastConnect**

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

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

For details for connecting Autonomous Data Warehouse with Oracle Cloud Infrastructure FastConnect see FastConnect Overview.
Loading Data into Autonomous Data Warehouse

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

Topics

• About Loading Data
• Load Data from Files in the Cloud
• Load Data Using Oracle Data Pump
• Replicate Data Using Oracle GoldenGate
• Load Data Using SQL*Loader

About Loading Data

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

In general you load data from files local to your client computer or from files stored in a cloud-based object store.

For the fastest data loading experience Oracle recommends uploading the source files to a cloud-based object store, such as Oracle Cloud Infrastructure Object Storage, before loading the data into your Autonomous Data Warehouse. Oracle provides support for loading files that are located locally in your data center, but when using this method of data loading you should factor in the transmission speeds across the Internet which may be significantly slower.

For more information on Oracle Cloud Infrastructure Object Storage, see Putting Data into Object Storage and Overview of Object Storage.

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 Database User Privileges for more information.

Load Data from Files in the Cloud

The PL/SQL package DBMS_CLOUD provides support for loading data from text files in the Cloud to tables created in your Autonomous Data Warehouse dedicated database.
Create Credentials, Create the Table, and Copy Data

For data loading from files in the Cloud, you need to first store your object storage credentials in your dedicated 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 Cloud Object Storage credential using the `DBMS_CREDENTIAL.CREATE_CREDENTIAL` procedure. For example:

   BEGIN
   DBMS_CREDENTIAL.CREATE_CREDENTIAL(
     credential_name => 'DEF_CRED_NAME',
     username => 'atpc_user@oracle.com',
     password => 'password'
   );
   END;

   The values you provide for `username` and `password` depend on the Cloud Object Storage service you are using:
   - **Oracle Cloud Infrastructure Object Storage**: `username` is your Oracle Cloud Infrastructure user name and `password` is your Oracle Cloud Infrastructure auth token. See Working with Auth Tokens.
   - **Oracle Cloud Infrastructure Object Storage Classic**: `username` is your Oracle Cloud Infrastructure Classic user name and `password` is your Oracle Cloud Infrastructure Classic 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.

2. Create the table that will contain the data. For example:

   ```sql
   CREATE TABLE CHANNELS
   (channel_id CHAR(1),
    channel_desc VARCHAR2(20),
    channel_class VARCHAR2(20)
   );
   /
   ```

3. Load data into the table using the procedure `DBMS_CLOUD.COPY_DATA`. For example:

   BEGIN
   DBMS_CLOUD.COPY_DATA(
     table_name => 'CHANNELS',
     credential_name => 'DEF_CRED_NAME',
     file_uri_list => 'https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/idthydc0kinr/mybucket/channels.txt',
     format => 'json_object(delimiter', ',')
   );
   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.

In this example, `file_uri_list` is an Oracle Cloud Infrastructure Swift URI that specifies the `channels.txt` file in the `mybucket` bucket in the `us-phoenix-1` region. (`idtydc0kinr` is the object storage namespace in which the bucket resides.) For information about the supported URI formats, see Cloud Object Storage URI Formats.

- `format`: defines the options you specify to describe the format of the source file. For information about the format options you can specify, see Format Parameter.

For more detailed information, see COPY_DATA Procedure.

Monitor and Troubleshoot Data Loading

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

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

The `LOGFILE_TABLE` column shows the name of the table you can query to look at the log of a load operation. For example, the following query shows the log of the load operation:

```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.

---

**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.

---

## Load Data Using Oracle Data Pump

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

Data Pump Import lets you import data from Data Pump files residing on the Oracle Cloud Infrastructure Object Storage and Oracle Cloud Infrastructure Object Storage Classic. You can save your data to your Cloud Object Store and use Oracle Data Pump to load data to Autonomous Data Warehouse.

**Topics**

- Export Data from Your Existing Oracle Database
- Upload the Export Files to Cloud Object Storage
- Import Data Using Oracle Data Pump

## Export Data from Your Existing Oracle Database

First you use Oracle Data Pump Export to export your existing Oracle Database schemas. Then you use Oracle Data Pump Import to migrate them to Autonomous Data Warehouse.

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

```sql
exclude=cluster, db_link
parallel=n
schemas=schema name
dumpfile=export%u.dmp
```

Oracle Data Pump Export provides several export modes, Oracle recommends using the schema mode for migrating to Autonomous Data Warehouse. You can list the schemas you want to export by using the `schemas` parameter.
For a faster migration, export your schemas into multiple Data Pump files and use parallelism. You can specify the dump file name format you want to use with the `dumpfile` parameter. Set the `parallel` parameter to at least the number of CPUs you have in your Autonomous Data Warehouse database.

The `exclude` and `data_options` parameters ensure that the object types not available in Autonomous Data Warehouse are not exported and table partitions are grouped together so that they can be imported faster to Autonomous Data Warehouse.

The following example exports the SH schema from a source Oracle Database for migration to an Autonomous Data Warehouse database with 16 CPUs:

```bash
expdp sh/sh@orcl 
exclude=cluster, db_link 
parallel=16 
schemas=sh 
dumpfile=export%u.dmp
```

You can use other Data Pump Export parameters, such as `compression`, depending on your requirements. For more information on Oracle Data Pump Export see Oracle Database Utilities.

## Upload the Export Files to Cloud Object Storage

Before you can import the data you exported from the source Oracle Database, you need to upload the export files to cloud object storage.

You can upload the export files to an existing storage bucket in Oracle Cloud Infrastructure Object Storage or an existing storage container in Oracle Cloud Infrastructure Object Storage Classic. Or you use the following procedure to create a new storage bucket and upload the export files to it.

1. Sign in to your Oracle Cloud Account at cloud.oracle.com.
2. From the Oracle Cloud Infrastructure left navigation list choose Object Storage and then choose Object Storage from the sublist.
3. Pick a compartment to create the storage bucket in.
4. Click **Create Bucket**.
5. In the Create Bucket dialog, give the bucket a name and then click **Create Bucket**.
6. After the bucket is created, click its name in the list of buckets to display its Bucket Details page.
7. In the Objects box, click **Upload Objects**.
8. In the Upload Objects dialog, click the **select files** link in the Choose Files From Your Computer box.
9. In the file browser, navigate to and select your export files. Then, click **Open**.
10. In the Upload Objects dialog, click **Upload Objects** to start uploading the files you selected.
11. After the uploads complete, close the Upload Objects dialog.
Import Data Using Oracle Data Pump

Oracle recommends using the latest Oracle Data Pump version for importing data from Data Pump files into your Autonomous Data Warehouse as it contains enhancements and fixes for a better experience.

Download the latest version of the Oracle Instant Client Basic Package and 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.

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.

Data Pump Import versions 12.2.0.1 and earlier do not have the credential parameter. If you are using an older version of Data Pump Import you need to define a default credential property for Autonomous Data Warehouse and use the default_credential keyword in the dumpfile parameter.

In Oracle Data Pump, if your source files reside in Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure native URIs or the Swift URIs. See Cloud Object Storage URI Formats for details on these URI formats.

1. Store your Cloud Object Storage credential using the DBMS_CREDENTIAL.CREATE_CREDENTIAL procedure. For example:

   ```sql
   BEGIN
   DBMS_CREDENTIAL.CREATE_CREDENTIAL(
     credential_name => 'DEF_CRED_NAME',
     username => 'atpc_user@oracle.com',
     password => 'password'
   );
   END;
   /
   
   The values you provide for username and password depend on the Cloud Object Storage service you are using:
   
   - **Oracle Cloud Infrastructure Object Storage**: username is your Oracle Cloud Infrastructure user name and password is your Oracle Cloud Infrastructure auth token. See Working with Auth Tokens.
   
   - **Oracle Cloud Infrastructure Object Storage Classic**: username is your Oracle Cloud Infrastructure Classic user name and password is your Oracle Cloud Infrastructure Classic password.

2. If you are using Oracle Data Pump version 12.2.0.1 or earlier, set the credential as the default credential for your Autonomous Data Warehouse, as the ADMIN user. For example:

   ```sql
   alter database property set default_credential = 'ADMIN.DEF_CRED_NAME'
   
3. Run Data Pump Import with the dumpfile parameter set to the list of file URLs on your Cloud Object Storage.
Note:

In the following examples, the `nologfile=yes` option is specified. This option is required for Autonomous Data Warehouse dedicated databases.

- **Oracle Data Pump version 18.3 or later**: set the `credential` parameter to the name of the credential you created in Step 1. For example:

  ```
  impdp admin/password@ATPC1_high 
  credential=def_cred_name 
  dumpfile=https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/idthydc0kinr/mybucket/export%u.dmp 
  parallel=16 
  transform=segment_attributes:n 
  transform=dwcs_cvt_iots:y 
  transform=constraint_use_default_index:y 
  exclude=cluster, db_link 
  nologfile=yes
  ```

  In this example, `dumpfile` is an Oracle Cloud Infrastructure Swift URI that specifies all files whose name matches `export<number>.dmp` in the `mybucket` bucket in the `us-phoenix-1` region. (`idthydc0kinr` is the object storage namespace in which the bucket resides.)

- **Oracle Data Pump version 12.2.0.1 or earlier**: start the value of the `dumpfile` parameter with the `default_credential` keyword and a colon. For example:

  ```
  impdp admin/password@ATPC1_high 
  dumpfile=default_credential:https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/idthydc0kinr/mybucket/export%u.dmp 
  parallel=16 
  transform=segment_attributes:n 
  exclude=cluster, db_link 
  nologfile=yes
  ```

  For the best import performance use the high database service for your import connection and set the `parallel` parameter to the number of CPUs your database has.

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

  For information on disallowed objects in Autonomous Data Warehouse, see Limitations on the Use of SQL Commands.

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

Replicate Data Using Oracle GoldenGate

You can replicate data to Autonomous Data Warehouse using Oracle GoldenGate and Oracle GoldenGate Cloud Service.

For more information, see Replicating Data to Oracle Autonomous Database.
Load Data Using SQL*Loader

You can use Oracle SQL*Loader to load data from local files in your client machine into Autonomous Data Warehouse.

Using SQL*Loader may be suitable for loading small amounts of data, as the load performance depends on the network bandwidth between your client and Autonomous Data Warehouse. For large amounts of data Oracle recommends loading data from the Cloud Object Storage. (For information on loading from Cloud Object Store, see Load Data Using Oracle Data Pump).

Oracle recommends using the following SQL*Loader parameters for the best load performance:

readsize=100M
bindsize=100M
direct=N

For detailed information on SQL*Loader parameters see Oracle Database Utilities.

For loading multiple files at the same time you can invoke a separate SQL*Loader session for each file.

Autonomous Data Warehouse gathers optimizer statistics for your tables during bulk load operations if you use the recommended parameters. If you do not use the recommended parameters, then you need to gather optimizer statistics manually as explained in Manage Optimizer Statistics 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 Data Warehouse

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

External data is not managed by the database; however, you can use `DBMS_CLOUD` procedures to query your external data. Although queries on external data will not be as fast as queries on database tables, you can use this approach to quickly start running queries on your external source files and external data.

You can use `DBMS_CLOUD` procedures to validate the data in the external 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 Database User Privileges for more information.

Topics
- Query External Data
- Validate External Data
- View Logs for Data Validation

Query External Data

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

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

<table>
<thead>
<tr>
<th>Channel</th>
<th>Source</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Direct Sales</td>
<td>Direct</td>
</tr>
<tr>
<td>T</td>
<td>Tele Sales</td>
<td>Direct</td>
</tr>
<tr>
<td>C</td>
<td>Catalog</td>
<td>Indirect</td>
</tr>
<tr>
<td>I</td>
<td>Internet</td>
<td>Indirect</td>
</tr>
<tr>
<td>P</td>
<td>Partners</td>
<td>Others</td>
</tr>
</tbody>
</table>

1. Store your Cloud Object Storage credential using the `DBMS_CREDENTIAL.CREATE_CREDENTIAL` procedure. For example:

```sql
BEGIN
    DBMS_CREDENTIAL.CREATE_CREDENTIAL(
        credential_name => 'DEF_CRED_NAME',
```
username => 'atpc_user@oracle.com',
password => 'password'
);
END;
/

The values you provide for username and password depend on the Cloud Object Storage service you are using:

- **Oracle Cloud Infrastructure Object Storage**: username is your Oracle Cloud Infrastructure user name and password is your Oracle Cloud Infrastructure auth token. See Working with Auth Tokens.

- **Oracle Cloud Infrastructure Object Storage Classic**: username is your Oracle Cloud Infrastructure Classic user name and password is your Oracle Cloud Infrastructure Classic 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.

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://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/idthydc0kinr/mybucket/channels.txt',
      format => 'json_object("delimiter" value ","),
      column_list => 'CHANNEL_ID NUMBER,
                     CHANNEL_DESC VARCHAR2(20),
                     CHANNEL_CLASS VARCHAR2(20) ');
END;
/
```

The parameters are:

- **table_name**: is the external table name.
- **credential_name**: is the name of the credential created in the previous step.
- **file_uri_list**: is a comma delimitied list of the source files you want to query.

In this example, file_uri_list is an Oracle Cloud Infrastructure Swift URI that specifies the channels.txt file in the mybucket bucket in the us-phoenix-1 region. (idthydc0kinr is the object storage namespace in which the bucket resides.) For information about the supported URI formats, see Cloud Object Storage URI Formats.

- **format**: defines the options you can specify to describe the format of the source file. For information about the format options you can specify, see Format Parameter.
• **column_list**: is a comma delimited list of the column definitions in the source files.

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 file(s). 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 `format` parameter options 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.

---

**Validate External Data**

To validate an external table, you use the procedure `DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE`.

Before validating an external table you need to create the external table using the `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` procedure. Then use the `DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE` procedure to validate it. For example:

```sql
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 created 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:

```sql
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`.

### View Logs for Data Validation

After validating an external table, 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 tables to view load validation information. For example use this `SELECT` statement 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>CHANNELS_EXT</td>
<td>SH</td>
<td>VALIDATE</td>
<td>COMPLETED</td>
<td>13-NOV-17...</td>
<td>13-NOV-17...</td>
</tr>
<tr>
<td>VALIDATE$21_LOG</td>
<td>VALIDATE$21_BAD</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:

```
SELECT * FROM VALIDATE$21_LOG;
```
The column BADFILE_TABLE shows the name of the table you can query to look at the rows where there were errors during validation. For example, the following query shows the rejected records for the above validate operation:

```
SELECT * FROM VALIDATE$21_BAD;
```

Depending on the errors shown in the log and the rows shown in the BADFILE_TABLE, you can correct the error by dropping the external table using the DROP TABLE command and recreating it by specifying the correct format options in DBMS_CLOUD.CREATE_EXTERNAL_TABLE.

**Note:**

The LOGFILE_TABLE and BADFILE_TABLE tables are stored for two days for each validate operation and then removed automatically.
Using Analytics and Visualization

Gain insight into your data with and Oracle Analytics Desktop.

Working with Oracle Analytics Desktop in Autonomous Data Warehouse

Gain insight into your data with Oracle Analytics Desktop. Oracle Analytics Desktop lets you explore your Autonomous Data Warehouse data through interactive visualizations.

Oracle Analytics Desktop provides powerful personal data exploration and visualization in a simple per-user desktop download. Oracle Analytics Desktop is the perfect tool for quick exploration of sample data from multiple sources or for rapid analysis and investigation of your own local data sets.

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

Oracle Analytics Desktop also gives you a preview of the self-service visualization capabilities included in Oracle Analytics Cloud, Oracle's industrial-strength cloud analytics platform. Oracle Analytics Cloud extends the data exploration and visualization experience by offering secure sharing and collaboration across the enterprise, additional data sources, greater scale, and a full mobile experience including proactive self-learning analytics delivered to your device. Try Oracle Analytics Desktop for personal analytics and to sample a taste of Oracle's broader analytics portfolio.

Oracle Analytics Desktop’s benefits include:

- A personal, single-user desktop application.
- Offline availability.
- Completely private analysis.
- Full control of data source connections.
- Lightweight single-file download.
- No remote server infrastructure.
- No administration tasks.

For details on connecting Autonomous Data Warehouse with Oracle Analytics Desktop, see User’s Guide for Oracle Data Visualization Desktop.

To download Oracle Analytics Desktop, see Oracle Data Visualization Desktop.
Exporting Data to Other Oracle Databases

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

To export data and move the data from Autonomous Data Warehouse to other Oracle databases, do the following:

1. Use Data Pump Export to export to a directory on Autonomous Data Warehouse.
2. Move the dump file set from the directory on Autonomous Data Warehouse to your Cloud Object Store.
3. Depending on the target database you may need to download the dump files from the Cloud Object Store.
4. Run Data Pump Import with the dump files.
5. Perform any required clean up such as removing the dump file set from Cloud Object Store.

Topics

- Use Data Pump to Create a Dump File Set on Autonomous Data Warehouse
- Move Dump File Set from Autonomous Data Warehouse to Your Cloud Object Store
- Download Dump Files, Run Data Pump Import, and Clean Up Object Store

Use Data Pump to Create a Dump File Set on Autonomous Data Warehouse

Oracle recommends using the latest Oracle Data Pump version for exporting data from Autonomous Data Warehouse to other Oracle databases, as it contains enhancements and fixes for a better experience.

Download the latest version of Oracle Instant Client and download the Tools Package, which includes Oracle Data Pump, for your platform from Oracle Instant Client Downloads. See the installation instructions on the platform install download page for the installation steps required after you download Oracle Instant Client and the Tools Package.

1. Create a directory in which to store the dump files containing the exported data. For example:

   CREATE DIRECTORY data_export_dir as 'data_export';

2. 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 a dedicated database named ATPC1 with 16 OCPUs:

expdp sales/password@ATPC1_high
directory=data_export_dir
dumpfile=exp%U.dmp
parallel=16
encryption_pwd_prompt=yes
filesize=1G
logfile=export.log

**Note:**

If during the export with expdp you use the encryption_pwd_prompt=yes parameter, then you must use encryption_pwd_prompt=yes with your import and input the same password at the impdp prompt to decrypt the dump files (remember the password you supply with export). The maximum length of the encryption password is 128 bytes.

For the best export performance use the HIGH database service for your export connection and set the PARALLEL parameter to the number of OCPUs in your a dedicated database. For information on which database service name to connect to run Data Pump Export, see Predefined Database Service Names for Autonomous Data Warehouse Dedicated Databases.

After the export is finished you can see the generated dump files by running a query like the following:

```sql
SELECT * FROM DBMS_CLOUD.LIST_FILES('DATA_EXPORT_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-</td>
</tr>
<tr>
<td>exp02.dmp</td>
<td>8192</td>
<td>12-NOV-19 06.10.48.0 PM GMT</td>
<td>12-</td>
</tr>
<tr>
<td>exp03.dmp</td>
<td>1171456</td>
<td>12-NOV-19 06.10.48.0 PM GMT</td>
<td>12-</td>
</tr>
<tr>
<td>exp04.dmp</td>
<td>348160</td>
<td>12-NOV-19 06.10.48.0 PM GMT</td>
<td>12-</td>
</tr>
<tr>
<td>export.log</td>
<td>1663</td>
<td>12-NOV-19 06.10.50.0 PM GMT</td>
<td>12-</td>
</tr>
</tbody>
</table>
Move Dump File Set from Autonomous Data Warehouse to Your Cloud Object Store

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

1. Connect to your Autonomous Data Warehouse database.
2. Store your Cloud Object Storage credential using the DBMS_CREDENTIAL.CREATE_CREDENTIAL procedure. For example:

   BEGIN
   DBMS_CREDENTIAL.CREATE_CREDENTIAL(
     credential_name => 'DEF_CRED_NAME',
     username => 'atpc_user@oracle.com',
     password => 'password'
   );
   END;
   /

   The values you provide for username and password depend on the Cloud Object Storage service you are using:

   - **Oracle Cloud Infrastructure Object Storage**: username is your Oracle Cloud Infrastructure user name and password is your Oracle Cloud Infrastructure auth token. See Working with Auth Tokens.

   - **Oracle Cloud Infrastructure Object Storage Classic**: username is your Oracle Cloud Infrastructure Classic user name and password is your Oracle Cloud Infrastructure Classic 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.

3. Move the dump files from the Autonomous Data Warehouse dedicated database to your Cloud Object Store by calling DBMS_CLOUD.PUT_OBJECT.
For example:

BEGIN
  DBMS_CLOUD.PUT_OBJECT(credential_name => 'DEF_CRED_NAME',
                             object_uri => 'https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/idthydc0kinr/mybucket/exp01.dmp',
                             directory_name => 'DATA_EXPORT_DIR',
                             file_name => 'exp01.dmp');
  DBMS_CLOUD.PUT_OBJECT(credential_name => 'DEF_CRED_NAME',
                             object_uri => 'https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/idthydc0kinr/mybucket/exp02.dmp',
                             directory_name => 'DATA_EXPORT_DIR',
                             file_name => 'exp02.dmp');
  DBMS_CLOUD.PUT_OBJECT(credential_name => 'DEF_CRED_NAME',
                             object_uri => 'https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/idthydc0kinr/mybucket/exp03.dmp',
                             directory_name => 'DATA_EXPORT_DIR',
                             file_name => 'exp03.dmp');
  DBMS_CLOUD.PUT_OBJECT(credential_name => 'DEF_CRED_NAME',
                             object_uri => 'https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/idthydc0kinr/mybucket/exp04.dmp',
                             directory_name => 'DATA_EXPORT_DIR',
                             file_name => 'exp04.dmp');
END; /

See PUT_OBJECT Procedure for information on PUT_OBJECT.

Download Dump Files, Run Data Pump Import, and Clean Up Object Store

If required, download the dump files from Cloud Object Store and use Oracle Data Pump Import to import the dump file set to the target database. Then perform any required clean up.

1. Download the dump files from Cloud Object Store.

   Note:
   This step is not needed if you are importing the data to an Autonomous Data Warehouse database or to an Autonomous Transaction Processing database.

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

3. Perform post import clean up tasks. If you are done importing the dump files to your target database(s) 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.
Creating Applications with Oracle Application Express in Autonomous Database

You can create applications with Oracle Application Express on Autonomous Data Warehouse.

Topics

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

About Oracle Application Express

Oracle Application Express (APEX) is a low-code development platform that enables you to build scalable, secure enterprise applications with world-class features that can be deployed anywhere.

Oracle APEX provides you with an easy-to-use browser-based environment to load data, manage database objects, develop REST interfaces, and build applications which look and run great on both desktop and mobile devices. You can use Oracle APEX to develop a wide variety of solutions: import spreadsheets and develop a single source of truth in minutes, create compelling data visualizations against your existing data, deploy productivity applications to elegantly solve a business need, or build your next mission-critical data management application.

Oracle APEX embraces SQL. Anything you can express with SQL can be easily employed in an Oracle APEX application. Oracle APEX also embodies low code with powerful data management and data visualization components, as well as responsive development out of the box. Instead of writing code by hand, you are able to use intelligent wizards to guide you through the rapid creation of applications and components.

Oracle APEX on Autonomous Database provides a preconfigured, fully managed and secured environment to both build and deploy world-class data-centric applications. There are no limits on the number of developers or end users for your Oracle APEX applications; Autonomous Database can instantly scale compute and storage online as needed, based upon your workload. Additionally, Oracle APEX applications
developed on-premises can be easily deployed to Oracle APEX on Autonomous Database, or vice-versa.

Configuration, patching, monitoring, and upgrading of all Oracle Application Express components is fully managed by Oracle, leaving you free to focus on developing your solutions and solving your business problems. With Oracle APEX and low code, your organization can be more agile and develop solutions faster, for less cost, and with greater consistency. You can adapt to changing requirements with ease. And you can empower professional developers and everyone else in your organization to be a part of the solution.

This chapter covers information on Oracle Application Express specific to working on Autonomous Data Warehouse. See Oracle Application Express for complete information.

Access Oracle Application Express Administration Services

Each Autonomous Data Warehouse instance includes a dedicated instance of Oracle Application Express; you can use this instance to create multiple workspaces. A workspace is a shared work area where you can build applications. You create workspaces in Application Express Administration Services.

To access Application Express Administration Services:

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

1. On the instance details page click Service Console.
2. Click Development.
3. Click APEX.
The Application Express Administration Services sign-in page appears.

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

4. In the **Password** field, enter the password for the Autonomous Data Warehouse ADMIN user.
5. Click **Sign In to Administration**.

When you sign in for the first time, follow the prompts to create an Application Express workspace. See **Create Oracle Application Express Workspaces in Autonomous Data Warehouse** for more information.

You can also use Administration Services to manage your Application Express instance. See Oracle Application Express Administration Services in **Oracle Application Express Administration Guide** for more information.

---

### Create Oracle Application Express Workspaces in Autonomous Data Warehouse

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

To create an Oracle Application Express workspace:

1. **Sign in to Application Express Administration Services.**

   * See **Access Oracle Application Express Administration Services** for more information.

2. **Click Create Workspace.**
3. On the Create Workspace page, in the **Database User** field, enter a new database username or choose an existing user from the list.

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.

5. (optional) In the **Workspace Name** field, change the name of the workspace that was automatically populated.

6. Click **Create Workspace**.

**Access Oracle Application Express App Builder**

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

To access Oracle Application Express App Builder:

1. Sign in to Application Express using the workspace name, username, and password you specify when you create the workspace.

2. On the Workspace home page, click the App Builder icon.

You can create additional developer accounts in Application Express workspaces and provide direct access to the Application Express instance. When you create a developer account, a corresponding database user is automatically created.

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

1. Sign in to Application Express using the workspace name, username, and password you specified when you created the workspace.

2. Pull down the **Administration** menu in the upper right of any page and choose **Manage Users and Groups**.

3. Click **Create User**.
4. In the **Username** field, enter a username.

5. In the **Email Address** field, enter an email address.

6. (Optional) Use the on-screen and in-line help to fill in additional fields.

7. In the **User is a developer** field, select **Yes**.

8. In the **Password** field, enter a strong password.

9. In the **Confirm Password** field, confirm the password.

10. Click **Create User**.

To share sign-in details with developers:

1. Select an Autonomous Data Warehouse instance.

2. On the instance details page click **Service Console**.

3. Click **Development**.

4. Right-click **APEX** and choose **Copy URL**.

5. Provide the copied URL, along with the Workspace Name, the Username, and the Password for the developer account you created.

Using this URL developers can access the Application Express environment without having to navigate to the Autonomous Data Warehouse Service Console.

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

---

**Use Web Services with Oracle Application Express**

You can interact with both SOAP and RESTful style web services from Application Express in your Autonomous Data Warehouse instance.

Web services enable applications to interact with one another over the web in a platform-neutral, language independent environment. In a typical web services scenario, a business application sends a request to a service at a given URL by using the HTTP protocol. The service receives the request, processes it, and returns a response. Web services are typically based on Simple Object Access Protocol (SOAP) or Representational State Transfer (REST) architectures.

Using Web Source Modules, Application Express developers can declaratively access data services from a variety of REST endpoints, allowing both read and write operations. In addition to supporting smart caching rules for remote REST data, Oracle Application Express also offers the unique ability to directly manipulate the results of REST data sources using industry standard SQL.

The **APEX_WEB_SERVICE** package enables you to integrate other systems with Application Express by allowing you to interact with web services anywhere you can use PL/SQL in your application. The package contains procedures and functions to call both SOAP and RESTful style web services, and to simplify implementation of OAuth 2.0 flows.

Note the following when working with web services in Application Express with Autonomous Data Warehouse:

- All web services must be secured. Only HTTPS services are supported on the default port (443). Your Application Express instance is pre-configured with an Oracle Wallet that contains more than 90 of the most common trusted root and
intermediate SSL certificates. The APEX_WEB_SERVICE package automatically takes advantage of this Oracle Wallet without additional configuration from application developers. This Oracle Wallet is centrally managed and therefore you cannot consume 3rd party web services that are protected using self-signed SSL certificates.

- Each Autonomous Data Warehouse instance is preconfigured with a network access control list (ACL) to permit outbound web service calls from Application Express. No further configuration by application developers is necessary.
- Your Application Express instance does not require an outbound web proxy.
- There is a limit of 50,000 outbound web service requests per Application Express workspace in a 24-hour period.

To learn more, see:

- APEX_WEB_SERVICE in Oracle Application Express API Reference
- Managing Web Source Modules in Oracle Application Express App Builder User’s Guide

### Send Email from Oracle Application Express

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

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

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

1. Identify the SMTP connection endpoint for Email Delivery. You configure the endpoint as the SMTP Host in your Application Express instance in Step 4. See Configure SMTP Connection for more information.
2. Generate SMTP credentials for Email Delivery. Your Application Express instance uses credentials to authenticate with Email Delivery servers when you send email. See Generate SMTP Credentials for a User for more information.
3. Create an approved sender for Email Delivery. You need to complete this step for all email addresses you use as the "From" with APEX_MAIL.SEND calls, as the Application Email From Address in your apps, or in the SMTP_FROM instance parameter. See Managing Approved Senders for more information.
4. Connect to your Autonomous Data Warehouse as ADMIN user using a SQL client and configure the following SMTP parameters using APEX_INSTANCE_ADMIN.SET_PARAMETER:
   - SMTP_HOST_ADDRESS: Specifies the SMTP connection endpoint from Step 1.
   - SMTP_USERNAME: Specifies the SMTP credential user name from Step 2.
   - SMTP_PASSWORD: Specifies the SMTP credential password from Step 2.

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. Send a test email using APEX SQL Workshop, SQL Commands specifying one of
the approved senders from Step 3 as "From". For example:

BEGIN
  APEX_MAIL.SEND(p_from => 'alice@example.com',
                  p_to   => 'bob@example.com',
                  p_subj => 'Email from Oracle Autonomous Database',
                  p_body => 'Sent using APEX_MAIL');
END;
/

6. To monitor email delivery in your Application Express instance:
   a. Sign in to APEX Administration Services.
   b. Open the Manage Instance page.
   c. Click the Mail Queue link in the Manage Meta Data section.

   Alternatively, query APEX_MAIL_QUEUE and APEX_MAIL_LOG views using a SQL
   client.

   **Note:**
   There is a limit of 5,000 emails per workspace in a 24-hour period. Oracle
   Cloud Infrastructure Email Delivery may impose additional limitations.

   For more information, see:
   - Overview of the Email Delivery Service
   - APEX_MAIL in Oracle Application Express API Reference
   - APEX_INSTANCE_ADMIN in Oracle Application Express API Reference

Restrictions and Limitations for Oracle Application Express
with Autonomous Data Warehouse

This section lists the feature restrictions and limitations of Oracle Application Express
when used within the context of Autonomous Data Warehouse. Certain limitations are
required to protect the security and performance of your Oracle Application Express
environment.

- Application Express Administration Services: Certain Application Express instance
  configuration options are disabled. The following are examples of configuration
  options that have been predefined by Oracle and cannot be altered:
Authentication scheme used to access App Builder

Ability to submit and approve self-service workspace requests and change requests

Web proxy, Oracle Wallet, and print server configuration

Daily limits of outbound web service calls and email messages

An option to make insecure outbound web service calls

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

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

- Only SMTP Application Express instance parameters may be set using the APEX_INSTANCE_ADMIN package. Other utilities provided by APEX_INSTANCE_ADMIN are disabled. See APEX_INSTANCE_ADMIN in Oracle Application Express API Reference for more information.

- Vanity URLs or custom domain names are not supported.
Developing RESTful Services in Autonomous Database

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

Topics:
- About Oracle REST Data Services in Autonomous Database
- Develop Oracle REST Data Services with Autonomous Database
- Develop SODA for REST with Autonomous Database

About Oracle REST Data Services in Autonomous Database

Oracle REST Data Services (ORDS) makes it easy to develop modern REST interfaces for relational data in the Autonomous Data Warehouse dedicated database. A mid-tier Java application, ORDS maps HTTP(S) verbs (GET, POST, PUT, DELETE, etc.) to database transactions and returns any results formatted using JSON.

Note:
The Oracle REST Data Services (ORDS) application in Autonomous Data Warehouse is preconfigured and fully managed. It is not possible to connect a customer-managed ORDS application to Autonomous Data Warehouse.

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

Develop Oracle REST Data Services with Autonomous Database

Autonomous Data Warehouse supports Oracle REST Data Services (ORDS). Developing RESTful Services is easy with either of the following development interfaces:

- SQL Developer (desktop): With SQL Developer on your desktop, you can connect to your Autonomous Data Warehouse dedicated database and REST enable tables and views, or develop custom RESTful Services based on your SQL and PL/SQL code. See Connect with Oracle SQL Developer Version 18.2 or Later for more information.
Oracle Application Express (APEX): With APEX you can use the RESTful Services development pages to build and maintain your services and REST enabled objects. You can use the APEX SQL Workshop to access your Oracle RESTful Services and REST Enabled objects. See How to Access RESTful Services for more information.

The Autonomous Data Warehouse ADMIN account is REST Enabled. This allows for REST Services to be published in the ADMIN schemas and allows you to access SQL Developer Web using the ADMIN database user account. Oracle recommends you create an application schema account for your RESTful Services and REST Enabled objects. Services are secured using Database Authentication and your REST Enabled schema.

The authenticated database user is only permitted access if the schema is REST enabled and the URL mapping for the request points to their own schema, the user is not authenticated when a request points to any other database schema. For example, the following request authenticated as REST enabled schema HR, is accessible:

GET /ords/hr/module/service/

However, when authenticated as REST enabled schema SCOTT, the same request:

GET /ords/hr/module/service/

would result in an error: 401 HTTP Unauthorized response/error.

Any database user whose credentials are correct and meets the above rules is authenticated and granted the ORDS, mid-tier, role: SQL Developer. This 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.

Develop SODA for REST with Autonomous Database

Autonomous Data Warehouse supports Simple Oracle Document Access (SODA) for REST.

Simple Oracle Document Access (SODA) for REST is a pre-deployed REST service that can be used to store JSON documents in an Autonomous Data Warehouse dedicated database. SODA enables flexible, NoSQL-style application development without having to use SQL. With SODA, JSON documents are stored in named collections and managed using simple CRUD operations (create, read, update and delete). And while SQL isn't required, JSON stored in SODA collections is still fully accessible from SQL when needed. For example, an operational application may be fully built using SODA (without SQL) but then the data may be later analyzed using SQL from outside of the application. Autonomous Database SODA gives application developers the best of the NoSQL and SQL worlds - fast, flexible, and scalable application development without losing the ability to leverage SQL for analytics and reporting.

SODA for REST is deployed in ORDS under the following URL pattern:

/ords/schema/soda/latest/*
Where schema corresponds to the REST enabled database schema (for example, "admin").

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

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

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

These commands insert three JSON documents into the fruit collection:

```
> curl -X POST -u 'ADMIN:password' \
-H "Content-Type: application/json" --data '{"name":"orange", "count":42}' \
"https://rzsf8o3up2w8rzc-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/fruit"

{"items": [{"id":"6F7E5C60197E4C8A83AC7D7654E2E375"...

> curl -X POST -u 'ADMIN:password' \
-H "Content-Type: application/json" --data '{"name":"pear", "count":5}' \
"https://rzsf8o3up2w8rzc-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/fruit"

{"items": [{"id":"83714B1E2BBA4FFA93B109E1B585"...

> curl -X POST -u 'ADMIN:password' \
-H "Content-Type: application/json" --data '{"name":"apple", "count":12, "color":"red"}' \
"https://rzsf8o3up2w8rzc-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://rzsf8o3up2w8rzc-db.adb.us-phoenix-1.oraclecloudapps.com/ords/admin/soda/latest/fruit?action=query"

{
  "items": [
    {
      "id":"6F7E5C60197E4C8A83AC7D7654E2E375",
      "etag":"57215643953D7C858A7CB28E14BB48549178BE307D1247860AFAB2A958400E16",
      "lastModified":"2019-07-12T19:00:28.199666Z",
      "created":"2019-07-12T19:00:28.199666Z",
      "value": {"name":"orange", "count":42}
    }
  ]
```

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

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

This query returns three rows:

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

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

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

See SODA for REST HTTP Operations for information on the SODA for REST HTTP operations.
Creating and Managing Directories

Oracle Autonomous Database provides you the ability to create directories in your dedicated database where you can store files, much like you can do in an OS file system. This ability to store files is especially useful when exporting and import data.

Topics:

• Create a Directory
• Drop a Directory
• List the Contents of a Directory
• Copy Files Between Cloud Object Storage and a Directory

Create a Directory

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 Autonomous Data Warehouse dedicated database and also creates the file system directory.
- You can create a directory in the root file system to see all the files with the following commands:

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

After you create the `ROOT_DIR` directory, use the following command to list all files:

```
SELECT * FROM DBMS_CLOUD.list_files('ROOT_DIR');
```

To run `DBMS_CLOUD.LIST_FILES` with a user other than ADMIN you need to grant read privileges on the directory to that user. See **LIST_FILES Function** for more information.

- Space used by the directories you create and their contents is part of your dedicated database's data storage allocation.

---

Drop a Directory

Use the database **DROP DIRECTORY** command to drop a directory object.

For example, the following command drops the database directory object `staging`:

```
DROP DIRECTORY staging;
```

The **DROP DIRECTORY** command does not delete files in the directory. If you want to delete the directory and the files in the directory, first use the procedure `DBMS_CLOUD.DELETE_FILE` to delete the files. See **DELETE_FILE Procedure** for more information.

To drop a directory, you must have the **DROP ANY DIRECTORY** system privilege. The ADMIN user is granted the **DROP ANY DIRECTORY** system privilege. The ADMIN user can grant **DROP ANY DIRECTORY** system privilege to other users.

See **DROP DIRECTORY** for more information.
Notes:

- If you just want to drop the directory and you do not remove the files in the directory, after you drop the directory you can view all the files in the file system, including any files that were in the directory you dropped, as follows:

  CREATE OR REPLACE DIRECTORY ROOT_DIR AS '';

  Then list the contents of ROOT_DIR with the following command:

  SELECT * FROM DBMS_CLOUD.LIST_FILES('ROOT_DIR');

  To run DBMS_CLOUD.LIST_FILES with a user other than ADMIN you need to grant read privileges on the directory to that user. See LIST_FILES Function for more information.

- The DROP DIRECTORY command does not remove the underlying file system directory. The Autonomous Data Warehouse dedicated database manages the underlying file system directory; users do not remove the file system directory.

List the Contents of a Directory

Use the function DBMS_CLOUD.LIST_FILES to list the contents of a directory.

For example, to list the contents of the stage directory, run the following query:

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

To run DBMS_CLOUD.LIST_FILES with a user other than ADMIN you need to grant read privileges on the directory to that user. See LIST_FILES Function for more information.

Copy Files Between Cloud Object Storage and a Directory

Use the procedure DBMS_CLOUD.PUT_OBJECT to copy a file from a directory to cloud object storage. Use the procedure DBMS_CLOUD.GET_OBJECT to copy a file from cloud object storage to a directory.

For example, to copy a file from cloud object storage to the stage directory, run the following command:

BEGIN
  DBMS_CLOUD.GET_OBJECT(
    credential_name => 'DEF_CRED_NAME',
    object_uri => 'https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/idthydc0kinr/mybucket/cwallet.sso',
    directory_name => 'STAGE');
END;
In this example, `object_uri` is an Oracle Cloud Infrastructure Swift URI that specifies the `cwallet.sso` file in the `mybucket` bucket in the `us-phoenix-1` region. (`idthydc0kinr` is the object storage namespace in which the bucket resides.) For information about the supported URI formats, see Cloud Object Storage URI Formats.

To run `DBMS_CLOUD.GET_OBJECT` with a user other than ADMIN you need to grant write privileges on the directory to that user.

To run `DBMS_CLOUD.PUT_OBJECT` with a user other than ADMIN you need to grant read privileges on the directory to that user.

See GET_OBJECT Procedure and PUT_OBJECT Procedure for more information.
Managing Autonomous Data Warehouse Dedicated Databases

This section describes management operations you can perform on Autonomous Data Warehouse dedicated databases.

Topics

• Create an Autonomous Data Warehouse Dedicated Database
• View Details of an Autonomous Data Warehouse Dedicated Database
• Add CPU or Storage Resources to an Autonomous Data Warehouse Dedicated Database
• Remove CPU or Storage Resources from an Autonomous Data Warehouse Dedicated Database
• Stop an Autonomous Data Warehouse Dedicated Database
• Start an Autonomous Data Warehouse Dedicated Database
• Restart an Autonomous Data Warehouse Dedicated Database
• Clone an Autonomous Data Warehouse Dedicated Database
• Move an Autonomous Data Warehouse Dedicated Database to a Different Compartment
• Terminate an Autonomous Data Warehouse Dedicated Database

Create an Autonomous Data Warehouse Dedicated Database

Follow these steps to create a new Autonomous Data Warehouse dedicated database using the Oracle Cloud Infrastructure console.

1. Sign in to your Oracle Cloud Account at cloud.oracle.com.
2. Click the menu icon in the top corner to display the navigation menu, and then click Autonomous Data Warehouse.

The Autonomous Databases page is displayed, showing the list of autonomous databases in your current region and compartment.

3. If needed, switch to the region where you want to create the dedicated database. See Switching Regions for information on switching regions and working in multiple regions.

4. Click Create Autonomous Database.
5. Select or enter basic information for your new dedicated database:
   - **Compartment**: Select a compartment from the list of available compartments.
   - **Display Name**: Specify a user-friendly description or other information that helps you easily identify the resource. The display name does not have to be unique.
   - **Database Name**: Specify the database name; it must consist of letters and numbers only. The maximum length is 14 characters.

   **Note:**
   The same database name cannot be used for multiple Autonomous Databases in the same tenancy in the same region.

6. For the workload type, choose **Data Warehouse**.

7. For the infrastructure type, select **Dedicated Infrastructure** to deploy your database on a dedicated Exadata infrastructure.

8. Enter the following to configure your database:
   - **CPU Core Count**: Specify the number of CPU cores for your dedicated database.
   - **Storage (TB)**: Specify the storage to allocate to your dedicated database, in terabytes.

9. Choose a container database:
   - Select the compartment.
   - Select the High Availability database container.

   High Availability deploys a clustered database in a single Availability Domain and provides monthly uptime matching your SLA. See Oracle Cloud Infrastructure Service Level Agreement for more information on SLAs.

10. Set the password for the Admin database user in your new dedicated database.
    
    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 Database Users.

11. Click **Show Advanced Options** to select additional features.

    - 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.

12. Click **Create Autonomous Database**.

    Your display shows **Provisioning** until the new Autonomous Data Warehouse dedicated database is available.
View Details of an Autonomous Data Warehouse Dedicated Database

1. Sign in to your Oracle Cloud Account at cloud.oracle.com.
2. Click the menu icon in the top corner to display the navigation menu, and then click Autonomous Data Warehouse.

   The Autonomous Databases page is displayed, showing the list of autonomous databases in your current region and compartment.

3. If needed, switch to the region hosting the dedicated database.

   See Switching Regions for information on switching regions and working in multiple regions.

4. If needed, switch to the compartment hosting the dedicated database.

   See Compartments for information on using and managing compartments.

5. In the list of autonomous databases, select the name of the dedicated database you want.

   The Details page for the selected dedicated database is displayed.

Add CPU or Storage Resources to an Autonomous Data Warehouse Dedicated Database

1. Go to the Details page of the Autonomous Data Warehouse dedicated database you want to add CPU or storage resources to.

   For instructions, see View Details of an Autonomous Data Warehouse Dedicated Database.


3. On the Scale Up/Down page, select the change in resources for your scale request.

   • Click up arrow to select a value for CPU Core Count. The default is no change.
   • Click up arrow to select a value for Storage (TB). The default is no change.

4. Click Update to change your resources.

Remove CPU or Storage Resources from an Autonomous Data Warehouse Dedicated Database

1. Go to the Details page of the Autonomous Data Warehouse dedicated database you want to remove CPU or storage resources from.

   For instructions, see View Details of an Autonomous Data Warehouse Dedicated Database.
2. On the Details page, select **Scale Up/Down**.

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

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

---

### Stop an Autonomous Data Warehouse Dedicated Database

1. Go to the **Details** page of the Autonomous Data Warehouse dedicated database you want to stop.
   For instructions, see **View Details of an Autonomous Data Warehouse Dedicated Database**.

2. On the **Details** page, select **Actions** and then select **Stop**.

3. Click **Stop** to confirm.

---

**Note:**

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

- Tools are no longer able to connect to the database.
- In-flight database transactions and queries are stopped.
- CPU billing is halted based on full-hour cycles of usage.

---

### Start an Autonomous Data Warehouse Dedicated Database

1. Go to the **Details** page of the Autonomous Data Warehouse dedicated database you want to start.
   For instructions, see **View Details of an Autonomous Data Warehouse Dedicated Database**.

2. On the **Details** page, select **Actions** and then select **Start**.
   **Start** is only shown for a stopped database.

3. Click **Start** to confirm.

---

**Note:**

When an Autonomous Data Warehouse dedicated database is started, CPU billing is initiated based on full-hour cycles of usage.
Restart an Autonomous Data Warehouse Dedicated Database

1. Go to the Details page of the Autonomous Data Warehouse dedicated database you want to restart.
   
   For instructions, see View Details of an Autonomous Data Warehouse Dedicated Database.

2. On the Details page, select More Actions and then select Restart.

3. In the confirmation dialog, select Restart to confirm.

Your dedicated database is restarted; that is, it is stopped and then immediately started.

Clone an Autonomous Data Warehouse Dedicated Database

You can clone an Autonomous Data Warehouse dedicated database using the Oracle Cloud Infrastructure console or REST API.

For information on using the REST API, see Cloning an Autonomous Database.

Follow these steps to use the Oracle Cloud Infrastructure console.

1. Go to the Details page of the Autonomous Data Warehouse dedicated database you want to clone.
   
   For instructions, see View Details of an Autonomous Data Warehouse Dedicated Database.

2. On the Details page, select More Actions and then select Create Clone.

   Create Clone is only enabled only when the database's state is Available or Available Needs Attention.

3. On the Create Autonomous Database Clone page, enter the following information:

   - **Choose Clone Type**: choose Full Clone or Metadata Clone.
     
     - The full clone option creates a new database that includes all of the source database's metadata and data.
     
     - The metadata clone option creates a new database that includes the source database's metadata, but not the source database's data.

   - **Origin Database Name** This read-only field identifies the name of the source database.

   - **Basic information**:
     
     - 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. The display name does not have to be unique.
– **Database Name** Specify the database name; it must consist of letters and numbers only. The maximum length is 14 characters.

**Note:**

The same database name cannot be used for multiple Autonomous Databases in the same tenancy in the same region.

– **Autonomous Container Database** This read-only field identifies the autonomous container database in which the clone will be created.

• **Configure the database:**
  – **CPU core count** Specify the number of cores for your dedicated database.
  – **Storage (TB)** Specify the storage you wish to make available to your dedicated database, in terabytes.

For a **Full Clone**, the minimum storage that you can specify is the source database's actual used space rounded up to the next TB.

• **Create administrator credentials**
  – **Username** This is a read-only field.
  – **Password** Set the password for the ADMIN database user in the cloned dedicated database. 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 Database Users](#).

**Note:**

You must specify a different password from the one for the ADMIN database user in the source database; otherwise, the clone operation will fail.

– **Confirm password** Specify a value to confirm the password.

• **Show Advanced Options** (Optional)
  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.

4. Click **Create Autonomous Database Clone**.

On the Oracle Cloud Infrastructure console the **State** shows Provisioning... until the new dedicated database is available.

Note the following information about the clone operation:

• You cannot initiate a new clone operation on a dedicated database that is already being cloned until the ongoing operation completes.

• You can only clone a dedicated database to the same autonomous container database as the source database.

Note the following information about the newly cloned dedicated database:
• Optimizer statistics are copied from the source database to the cloned database. Then:
  – For full clones, loads into tables behave the same as loading into a table with statistics already in place.
  – For metadata clones, the first load into a table clears the statistics for that table and updates the statistics with the new load.

For more information on Optimizer Statistics, see Optimizer Statistics Concepts.

• Resource management rules changed by the user in the source database are carried over to the cloned database.

• Performance data for the time before the clone operation is not available in the cloned database.

Move an Autonomous Data Warehouse Dedicated Database to a Different Compartment

Shows you the steps to move an Autonomous Data Warehouse dedicated database to a different Oracle Cloud Infrastructure compartment.

Note:

• To move an Autonomous Data Warehouse dedicated database you must have the right to manage autonomous databases in the database’s current compartment and in the compartment you are moving it to.

• As soon as you move an Autonomous Data Warehouse dedicated 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.

1. Go to the Details page of the Autonomous Data Warehouse dedicated database you want to move.
   For instructions, see View Details of an Autonomous Data Warehouse Dedicated Database.

2. On the Details page, select More Actions and then select Move Resource.

3. In the Move Resource to a Different Compartment page, select the new compartment.

4. Click Move Resource.
   See Moving Database Resources to a Different Compartment for more information.
Terminate an Autonomous Data Warehouse Dedicated Database

Note:
Terminating an Autonomous Data Warehouse dedicated database permanently deletes it and removes all automatic backups. You cannot recover a terminated database.

1. Go to the Details page of the Autonomous Data Warehouse dedicated database you want to terminate.
   For instructions, see View Details of an Autonomous Data Warehouse Dedicated Database.

2. On the Details page, select More Actions and then select Terminate.

3. On the Terminate Database page enter the database name to confirm that you want to terminate the database.

4. Click Terminate Database.
Managing Database Users

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

Topics

• Create Database Users
• Remove Database Users
• Manage the ADMIN Database User
• Manage Database User Privileges
• Use Oracle Database Vault to Manage Database User Privileges

Create Database Users

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

• As the ADMIN user run the following SQL statements:

```
CREATE USER new_user IDENTIFIED BY password DEFAULT TABLESPACE tablespace_name;
GRANT CREATE SESSION TO new_user;
```

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

Note:

The administrator needs to provide the credentials wallet to the user `new_user`. See Connecting to Autonomous Data Warehouse.

Autonomous Data Warehouse requires strong passwords; the password you specify must meet the default password complexity rules.

• The password must be between 12 and 30 characters long and must include at least one uppercase letter, one lowercase letter, and one numeric character.

  Note, the password limit is shown as 60 characters in some help tooltip popups. Limit passwords to a maximum of 30 characters.

• The password cannot contain the `username`.

• The password cannot be one of the last four passwords used for the same `username`. 
• The password cannot contain the double quote (" ) character.
• The password must not be the same password that is set less than 24 hours ago.

To unlock a database user account, connect to your database as the ADMIN user and run the following command:

```sql
ALTER USER username IDENTIFIED BY password ACCOUNT UNLOCK;
```

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

## Remove Database Users

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

• As the ADMIN user run the following SQL statement:

```sql
DROP USER user_name CASCADE;
```

This removes `user_name` and the objects owned by that user.

**Note:**

This removes all `user_name` objects and the data owned by `user_name` is deleted.

## Manage the ADMIN Database User

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

### Topics

- Change the ADMIN Database User Password
- Unlock the ADMIN Database User Account

## Change the ADMIN Database User Password

You change the ADMIN database user's password using the Oracle Cloud Infrastructure console.

1. Go to the **Details** page of the Autonomous Data Warehouse dedicated database whose ADMIN user password you want to change.
   
   For instructions, see View Details of an Autonomous Data Warehouse Dedicated Database.

2. On the **Details** page, select **More Actions** and then select **Admin Password**.

3. On the **Admin Password** page enter the new password and confirm.
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 Database Users.

4. Click Update.

Unlock the ADMIN Database User Account

To unlock the ADMIN database user account, change the ADMIN user's password using the Oracle Cloud Infrastructure console.

1. Go to the Details page of the Autonomous Data Warehouse dedicated database whose ADMIN user password you want to change.
   
   For instructions, see View Details of an Autonomous Data Warehouse Dedicated Database.

2. On the Details page, select More Actions and then select Admin Password.

3. On the Admin Password page enter the new password and confirm.

   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 Database Users.

4. Click Update.

Manage Database User Privileges

Autonomous Data Warehouse databases come with a predefined database role named DWROLE. This role provides the common privileges for a database developer or data scientist to performance real-time analytics.

To grant DWROLE role to your developers, connect to the database as the ADMIN user using any SQL client tool.

• As the ADMIN user run the following SQL statement:

   GRANT DWROLE TO user;

The privileges in DWROLE are the following:

CREATE ANALYTIC VIEW
CREATE ATTRIBUTE DIMENSION
ALTER SESSION
CREATE HIERARCHY
CREATE JOB
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
You can also grant individual privileges to users with the GRANT command instead of or in addition to granting DWROLE privileges. See Oracle Database SQL Language Reference.

**Note:**

Granting DWROLE to a user also grants UNLIMITED TABLESPACE to the user which means the user can use any amount of storage and this also overrides all explicit tablespace quotas for the user. If you want to limit the amount of storage for a user, do not grant DWROLE to that user.

---

**Use Oracle Database Vault to Manage Database User Privileges**

Oracle Autonomous Database supports Oracle Database Vault, which you can use to implement powerful security controls within your dedicated 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.

You can deploy controls to block privileged account access to application data and control sensitive operations inside the database. Trusted paths can be used to add additional security controls to authorized data access and database changes. Through the runtime analysis of privileges and roles, you can increase the security of existing applications by implementing least privileges and reducing the attack profile of your database accounts. Oracle Database Vault secures existing database environments transparently, eliminating costly and time consuming application changes.

Before using Oracle Database Vault, be sure to review What to Expect After You Enable Oracle Database Vault to gain an understanding of the impact of configuring and enabling Database Vault.

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

**Configure and Enable Oracle Database Vault**

To configure and enable Oracle Database Vault in your Autonomous Data Warehouse dedicated database, you create database users to act as the Database Vault owner and account manager, and then use these users when configuring and enabling Database Vault.

1. Connect as the ADMIN user to your dedicated database.
2. Create the Database Vault owner and account manager users; for example:
   ```sql
   create user dbv_owner identified by <password>:
   grant create session to dbv_owner;
   create user dbv_acctmgr identified by <password>:
   grant create session to dbv_acctmgr;
   ```
3. Configure Database Vault, providing the owner and account manager user names in the command; for example:
exec dvsys.configure_dv('dbv_owner','dbv_acctmgr');

4. Connect as the Database Vault owner (dbv_owner in this example) to your dedicated database.

5. Enable Database Vault:
   exec dbms_macadm.enable_dv;

6. Restart (that is, stop and then start) your Autonomous Data Warehouse dedicated deployment.
Managing and Monitoring Performance

This section describes managing and monitoring the performance of Autonomous Data Warehouse.

Topics
- Use SQL Developer Web to Manage and Monitor Databases
- Use Enterprise Manager to Manage and Monitor Databases
- Use Performance Hub to Monitor Databases
- Use Events and Notifications to Watch Resource Activity
- Use Work Requests to Monitor Long-Running Operations
- Manage Database Service Consumer Groups
- Manage Optimizer Statistics
- Manage Automatic Indexing

Use SQL Developer Web to Manage and Monitor Databases

Oracle SQL Developer Web, a browser-based application that provides many database development, management and monitoring features, is built into Autonomous Data Warehouse dedicated databases. Therefore, you can use it without downloading or installing additional software on your system.

Use SQL Developer Web monitoring features, like Performance Hub, to view and analyze performance of your database. To connect to SQL Developer Web, see Connect to Oracle SQL Developer Web in the Database. For information about the wide-ranging features of SQL Developer Web, see Using Oracle SQL Developer Web.

Use Enterprise Manager to Manage and Monitor Databases

You can use Oracle Enterprise Manager to manage and monitor Autonomous Data Warehouse dedicated databases.

Note:
You must be using, at a minimum, Enterprise Manager version 13.3 with EM DB Plugin Bundle Patch 13.3.2.0.190731 to manage and monitor dedicated databases.

Enterprise Manager is Oracle's management platform, providing a single pane of glass for managing all of your Oracle deployments, whether in your data centers or in Oracle
Cloud. Through deep integration with Oracle’s product stack, Enterprise Manager provides market-leading management and automation support for Oracle applications, databases, middleware, hardware and engineered systems.

To use Enterprise Manager to manage and monitor your dedicated database, you perform these high-level tasks:

1. Configure connectivity between your Enterprise Manager deployment (whether on-premises or on Oracle Cloud Infrastructure Marketplace) and your Autonomous Data Warehouse dedicated database.

2. Use the Enterprise Manager console, CLI or REST API to discover the dedicated database and add it as a target.

For detailed steps to perform these tasks, see Discover Autonomous Databases in Enterprise Manager Cloud Control Administrator’s Guide for Oracle Autonomous Databases.

After performing the discovery tasks, you can use Enterprise Manager to:

• Monitor the health and performance of your database and perform deep diagnostics on the Performance Hub.

• Perform database administration tasks such as storage management, and schema management tasks such as creating database objects.

For details about how to use these features, see Monitoring and Administration Tasks in Enterprise Manager Cloud Control Administrator’s Guide for Oracle Autonomous Databases.

Use Performance Hub to Monitor Databases

You can use Performance Hub to view real-time and historical performance data for an Autonomous Data Warehouse dedicated database. Performance Hub shows Active Session History (ASH) Analytics and SQL Monitoring.

1. Go to the Details page of the Autonomous Data Warehouse dedicated database you want to monitor with Performance Hub.

   For instructions, see View Details of an Autonomous Data Warehouse Dedicated Database.

2. On the Details page, select Performance Hub.

The Performance Hub page is displayed.
The Time Range selector is on the top of the Performance Hub page. Use the Select Duration field to set the time duration. By default, Last hour is selected. You can choose to view Last 8 hours, Last 24 hours, Last week, or specify a custom time range using the Custom option.

The Time Range field shows active sessions in chart form for the time period selected. The active sessions chart displays the average number of active sessions broken down by CPU, User I/O, and Wait. The active sessions chart also shows the Max CPU usage.

The sliding box on the time range chart is the time slider. Use the time slider to select the exact period of time for which data is displayed in the Performance Hub tables and graphs. This is a subsection of the period of time shown in the Time Range field.
You can slide the box to the left or the right to shift the time period under analysis. To slide the entire box, left-click anywhere inside the box and drag the box to the left or the right. Widen or narrow the box to increase or decrease the length of time under analysis. To widen or narrow the box, left-click and hold the handlebar on either side of the box, then drag to the left or the right to increase or decrease the size of the current time range box.

Click **Refresh** to refresh the data in Performance Hub according to the time range chosen.

Select either **ASH Analytics** or **SQL Monitoring**:

- **Active Session History (ASH) Analytics**:
  This 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.

- **SQL Monitoring**:
  The SQL statements are only monitored if they've been running for at least five seconds or if they're run in parallel. The table displays monitored SQL statement executions by dimensions including **Last Active Time**, **CPU Time**, and **Database Time**. The table displays currently running SQL statements and SQL statements that completed, failed, or were terminated. The columns in the table provide information for monitored SQL statements including **Status**, **Duration**, and **SQL ID**.

  The **Status** column has the following icons:
  - A spinning icon indicates that the SQL statement is executing.
  - A green check mark icon indicates that the SQL statement completed its execution during the specified time period.
  - A red cross icon indicates that the SQL statement did not complete, either due to an error, or due to the session being terminated.
  - A clock icon indicates that the SQL statement is queued.

To terminate a running or queued SQL statement, click **Kill Session**.

Select the link in the **SQL ID** column to go to the corresponding **Real-time SQL Monitoring** page. This page provides additional details to help you tune the selected SQL statement.

See **Active Session History (ASH)** in *Oracle Database Concepts* for more information on Active Session History.

---

**Use Events and Notifications to Watch Resource Activity**

When Oracle Autonomous Database performs operations on Autonomous Database resources, it sends *events* to the Events service. Using the Events service, you can create rules to capture these events and perform actions such as sending you emails using the Notifications service.

For more information about how the Events service works and how to set up rules and actions, see **Overview of Events**.
For a listing of the operations on Autonomous Database resources that generate events, see Autonomous Database Event Types.

Use Work Requests to Monitor Long-Running Operations

Oracle Autonomous Database is integrated with the Oracle Cloud Infrastructure Work Requests feature. Work requests allow you to monitor long-running operations like the creation of an Autonomous Database. A work request is an activity log that enables you to track each step in the operation's progress. Each work request has an OCID that allows you to interact with it programmatically and use it for automation.

For a list of Oracle Autonomous Database operations that create work requests, see Work Requests Integration.

For general information on using work requests in Oracle Cloud Infrastructure, see Work Requests and Work Requests API.

Manage Database Service Consumer Groups

Oracle Autonomous Data Warehouse uses Oracle Database Resource Manager consumer groups to define different workload characteristics for each of the pairs of database services described in Predefined Database Service Names for Autonomous Data Warehouse Dedicated Databases. If the default values used in these consumer groups do not meet a given database’s usage model, you can change the values.

Topics
- Manage CPU/IO Shares
- Manage Runaway SQL Statements

Manage CPU/IO Shares

Autonomous Data Warehouse comes with predefined CPU/IO shares assigned to different consumer groups. You can modify these predefined CPU/IO shares if your workload requires different CPU/IO resource allocations.

By default, the CPU/IO shares assigned to the consumer groups HIGH, MEDIUM, and LOW are 4, 2, and 1, respectively. The shares determine how much CPU/IO resources a consumer group can use with respect to the other consumer groups. With the default settings the consumer group HIGH will be able to use 4 times more CPU/IO resources compared to LOW, when needed, and 2 times more CPU/IO resources compared to MEDIUM, when needed.

You can set CPU/IO shares using the PL/SQL package cs_resource_manager.update_plan_directive. For example, running the following script as the ADMIN user sets CPU/IO shares to 8, 4, and 1 for consumer groups HIGH, MEDIUM, and LOW respectively. This will allow the consumer group HIGH to use 2 times more CPU/IO resources compared to the consumer group MEDIUM and 8 times CPU/IO resources compared to the consumer group LOW:

BEGIN
  cs_resource_manager.update_plan_directive(consumer_group => 'HIGH', shares => 8);
  cs_resource_manager.update_plan_directive(consumer_group => 'MEDIUM', shares => 4);
  cs_resource_manager.update_plan_directive(consumer_group => 'LOW', shares => 1);
END;
Manage Runaway SQL Statements

Use the PL/SQL procedure `cs_resource_manager.update_plan_directive` to change the rules Autonomous Data Warehouse uses to terminate SQL statements automatically for a given consumer group.

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.

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:

```sql
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:

```sql
BEGIN
    cs_resource_manager.update_plan_directive(consumer_group => 'HIGH',
    io_megabytes_limit => null, elapsed_time_limit => null);
END;
/
```

Manage Optimizer Statistics

Describes Autonomous Data Warehouse commands to run when you need to gather optimizer statistics or enable optimizer hints.

Managing Optimizer Statistics

Autonomous Data Warehouse gathers optimizer statistics automatically 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 Data Warehouse and runs in a standard maintenance window.
Managing Optimizer Hints

Autonomous Data Warehouse 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

The automatic indexing feature in Oracle Database is enabled in Autonomous Data Warehouse.

Creating indexes manually requires deep knowledge of the data model, application, and data distribution. Often DBAs make choices about which indexes to create, and then never revise their choices. As a result, opportunities for improvement are lost, and unnecessary indexes can become a performance liability.

With automatic indexing the database monitors the application workload, creating and maintaining indexes automatically. The indexing feature is implemented as an automatic task that runs at a fixed interval. You can control behavior at the statement level by using the `USE_AUTO_INDEXES` or `NO_USE_AUTO_INDEXES` hints. Automatic indexing hints enable the optimizer to use automatic indexes for SQL statements. You can use the `DBMS_AUTO_INDEX` package to report on the automatic task and to set your preferences.

For more information see Managing Auto Indexes in *Oracle Database Administrator's Guide*. 

Note:

For more information on maintenance window times and automatic optimizer statistics collection, see *Oracle Database Administrator's Guide*.

For more information on optimizer statistics see *Oracle Database SQL Tuning Guide*. 
Backing Up and Restoring Autonomous Data Warehouse

This section describes backup and recovery tasks on Autonomous Data Warehouse.

Topics
- About Backup and Recovery on Autonomous Data Warehouse
- Restore and Recover Your Autonomous Data Warehouse Dedicated Database
- Back Up Your Autonomous Data Warehouse Dedicated Database Manually

About Backup and Recovery on Autonomous Data Warehouse

Autonomous Data Warehouse automatically backs up your database for you. The retention period for backups is 60 days. You can restore and recover your database to any point-in-time in this retention period.

Manual Backups

You do not have to do any manual backups for your database as Autonomous Data Warehouse backs up your database automatically. You can do manual backups using the cloud console; for example if you want to take a backup before a major change to make restore and recovery faster. When you initiate a point-in-time recovery Autonomous Data Warehouse decides which backup to use for faster recovery.

Recovery

You can initiate recovery for your Autonomous Data Warehouse database using the cloud console. Autonomous Data Warehouse automatically restores and recovers your database to the point-in-time you specify.

Listing Backups

The list of backups available for recovery is shown on the Autonomous Database Details page under Backups.

Restore and Recover Your Autonomous Data Warehouse Dedicated Database

To restore and recover your database to a point in time, do the following:

1. Go to the Details page of the Autonomous Data Warehouse dedicated database you want to restore and recover.
For instructions, see View Details of an Autonomous Data Warehouse Dedicated Database.

2. On the Details page, select Actions and then select Restore.

3. In the Restore prompt, select Specify Timestamp or Select Backup to restore to a point in time or to restore from a specified backup.
   - SPECIFY TIMESTAMP: Enter a timestamp to restore to in the ENTER TIMESTAMP calendar field.
   - SELECT BACKUP: Select a backup from the list of backups. Limit the number of backups you see by specifying a period using the FROM and TO calendar fields.

4. Click Restore.

Note:

- Restoring a database puts it in the unavailable state during the restore operation. You cannot connect to the database in this state. The only lifecycle management operation supported in unavailable state is terminate.

The Details page shows Lifecycle State: Restore In Progress...

5. When the restore operation finishes your database is opened in the same state as before restoration.

Note:

- After restoring your database, all backups between the date the restore completes and the date you specified for the restore operation - the restore time - are invalidated. You cannot initiate further restore operations to any point in time between the restore time and restore completion time. You can only initiate new restore operations to a point in time older than the restore time or more recent than the time when the actual restore succeeded.

Back Up Your Autonomous Data Warehouse Dedicated Database Manually

In addition to automatic backups, Autonomous Data Warehouse also allows you take manual backups.

1. Go to the Details page of the Autonomous Data Warehouse dedicated database you want to manually back up.
   For instructions, see View Details of an Autonomous Data Warehouse Dedicated Database.


3. In the Create Manual Backup dialog enter a name in the Name field.
4. In the **Create Manual Backup** dialog click **Create**.

### Note:

While backing up a database, the database is fully functional; however during the backup lifecycle management operations are not allowed. For example, stopping the database is not allowed during the backup.
Cloud Object Storage URI Formats

The URI format to use when specifying source files in cloud object storage services depends on the object storage service you are using:

- Oracle Cloud Infrastructure Object Storage
- Oracle Cloud Infrastructure Object Storage Classic

Oracle Cloud Infrastructure Object Storage URI Formats

Note:

Autonomous Data Warehouse supports only standard-tier storage buckets. It does not support archive-tier storage buckets.

If your source files reside in Oracle Cloud Infrastructure Object Storage, you can use Oracle Cloud Infrastructure native URIs, Swift URIs, or pre-authenticated request URIs:

- Native URI format:
  https://objectstorage.region.oraclecloud.com/n/object-storage-namespace/b/bucket/o/filename

  Tip:
  You can view the Native URI of an object (such as an uploaded file) in its Object Details dialog:
  1. Go to the Bucket Details page of the bucket containing the object.
  2. In the bucket's list of objects, click the action menu icon and choose View Object Details.
     The URL Path (URI) field displays the object's Native URI value.

- Swift URI format:
  https://swiftobjectstorage.region.oraclecloud.com/v1/object-storage-namespace/b/bucket/filename

- Pre-Authenticated Request URI format:
  https://objectstorage.region.oraclecloud.com/p/encrypted-string/n/object-storage-namespace/b/bucket/o/filename

  For information about creating pre-authenticated request URIs, see Using Pre-Authenticated Requests in Oracle Cloud Infrastructure Documentation.
Tip:
You can view the `object-storage-namespace` value for a bucket in the `Namespace` field of the bucket's `Bucket Details` page.

Here are example URIs for the `myfile.dmp` file in the `mybucket` bucket in the `us-phoenix-1` region. (`idthydc0kinr` is the object storage namespace.)

- **Native URI format:**
  
  https://objectstorage.us-phoenix-1.oraclecloud.com/n/idthydc0kinr/b/mybucket/o/myfile.dmp

- **Swift URI format:**
  
  https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/idthydc0kinr/mybucket/myfile.dmp

- **Pre-Authenticated Request URI format:**
  
  https://objectstorage.us-phoenix-1.oraclecloud.com/p/zcH0IpixjxUv03nj7Kg6pb-iMJL-SdNc6FbRl9GSsk/n/idthydc0kinr/b/mybucket/o/myfile.dmp

**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](#).
Using Oracle Database Features in Autonomous Data Warehouse Dedicated Deployments

This appendix provides information on using Oracle Database features and options in Autonomous Data Warehouse dedicated deployments.

For equivalent information in other kinds of Autonomous Database deployments, see:

- For Autonomous Data Warehouse shared deployments: Autonomous Data Warehouse for Experienced Oracle Database Users
- For Autonomous Transaction Processing shared deployments: Autonomous Transaction Processing for Experienced Oracle Database Users
- For Autonomous Transaction Processing dedicated deployments: Using Oracle Database Features in Dedicated Autonomous Transaction Processing Deployments

Topics

- About the Autonomous Data Warehouse Dedicated Database
- Latest Autonomous Data Warehouse Database Features
- The ADMIN User and the SYS User
- Modifying Database Initialization Parameters
- Oracle Database Features That Are Not Supported
- Oracle Database Features with Limited Support
- Limitations on the Use of SQL Commands
- Limitations on the Use of Certain Data Types
- Limitations on the Use of Manually Created External Tables

About the Autonomous Data Warehouse Dedicated Database

Autonomous Data Warehouse configures and optimizes your database for you. You do not need to perform administration operations for configuring the database. SQL commands used solely for database administration are not available in this service. Similarly, other administrative interfaces and utilities such as RMAN are not available.

Characteristics of an Autonomous Data Warehouse dedicated database include:

- The default data and temporary tablespaces for the database are configured automatically.
- The name of the default data tablespace is DATA.
• The database character set is Unicode AL32UTF8.
• Hybrid Columnar Compression is enabled by default for all tables. You can specify different compression methods for your tables using the `table_compression` clause in your `CREATE TABLE` or `ALTER TABLE` commands.

Accessing the Autonomous Data Warehouse dedicated database:

• You do not have direct access to the database node or the local file system.
• You do not have access to the `SYSTEM` or `SYSAUX` tablespaces.

Parallel Execution:

• Parallel execution (parallelism within a SQL statement) is enabled by default. Degree of parallelism for SQL statements is set based on the number of OCPUs in the system and the database service you use when connecting to the database.
• Parallel DML is enabled by default. If you do not want to run DML operations in parallel you can disable parallel DML in your session using the following SQL command:

  ```sql
  ALTER SESSION DISABLE PARALLEL DML;
  ````

See Oracle Database 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. Change the `parallel_clause` value to `NOPARALLEL` or set the `PARALLEL` attribute to 1 to specify serial execution:

  ```sql
  ALTER INDEX index_name NOPARALLEL;
  ````

  or

  ```sql
  ALTER INDEX index_name PARALLEL 1;
  ````

Latest Autonomous Data Warehouse Database Features

Autonomous Data Warehouse includes the latest Oracle Database features.

Autonomous Data Warehouse includes features that:

• Automate index management tasks, such as creating, rebuilding, and dropping indexes based on changes in the application workload. See Managing Auto Indexes in Oracle Database Administrator’s Guide for more information.
• Gather real-time statistics automatically while a conventional DML workload is running. Because statistics can go stale between `DBMS_STATS` 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. You can manage and access statistics for conventional DML through PL/SQL packages, data dictionary views, and hints. See Real-Time Statistics in Oracle Database SQL Tuning Guide 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. See Configuring High-Frequency Automatic Optimizer
Statistics Collection in Oracle Database SQL Tuning Guide for more information.

• Quarantine execution plans for SQL statements that are terminated by the
  Resource Manager for consuming excessive system resources in an Oracle
database. You can configure quarantine settings for a SQL statement by
specifying limits on its resource consumption using procedures in the DBMS_SQLQ
package. When the SQL statement crosses any of these resource consumption
limits, it is terminated and the execution plan is quarantined. In this way, the
database prevents high-resource SQL statements from executing repeatedly. See
Quarantine for Execution Plans for SQL Statements Consuming Excessive
System Resources in Oracle Database Administrator’s Guide for more information.

• Optimize high-frequency single-row inserts for applications, such as Internet of
  Things (IoT) applications. See Enabling High Performance Data Streaming With
the Memoptimized Rowstore in Oracle Database Performance Tuning Guide.

The ADMIN User and the SYS User

In Oracle Autonomous Database the predefined administrative user is ADMIN. In
Oracle Database the predefined administrative user is SYS. While these two users
serve the same purpose in their respective databases, they are not the same and do
not have the same set of privileges.

Because Oracle Autonomous Database imposes security controls and performs
administrative database tasks for you, the ADMIN user does not have as many
privileges as the SYS user. Here is a list of the privileges that the ADMIN user does
not have but that the SYS user in an Oracle Database does have:

ALTER LOCKDOWN PROFILE
BACKUP ANY TABLE
BECOME USER
CREATE ANY JOB
CREATE ANY LIBRARY
CREATE LIBRARY
CREATE LOCKDOWN PROFILE
CREATE PLUGGABLE DATABASE
DEQUEUE ANY QUEUE
DROP LOCKDOWN PROFILE
EM EXPRESS CONNECT
ENQUEUE ANY QUEUE
EXPORT FULL DATABASE
FLASHBACK ANY TABLE
FLASHBACK ARCHIVE ADMINISTER
GRANT ANY PRIVILEGE
GRANT ANY ROLE
IMPORT FULL DATABASE
INHERIT ANY PRIVILEGES
LOGMINING
MANAGE ANY FILE GROUP
MANAGE ANY QUEUE
MANAGE FILE GROUP
USE ANY JOB RESOURCE
USE ANY SQL TRANSLATION PROFILE
Modifying Database Initialization Parameters

Autonomous Data Warehouse configures database initialization parameters automatically when you provision a database. You do not need to set any initialization parameters to start using your service. However, 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_AUTOFRESH_ENABLED
- CURRENT_SCHEMA (Session only by using ALTER SESSION)
- CURSOR_SHARING
- DB_BLOCK_CHECKING
- DDL_LOCK_TIMEOUT
- FIXED_DATE
- GLOBAL_NAMES
- HEAT_MAP
- MAX_IDLE_TIME
- NLSCALENDAR
- NLS_COMP
- NLS_CURRENCY
- NLS_DATE_FORMAT
- NLS_DATE_LANGUAGE
- NLS_DUAL_CURRENCY
- NLS_ISO_CURRENCY
- NLS_LENGTH_SEMANTICS
- 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 (Session only by using ALTER SESSION)
- OPTIMIZER_IGNORE_HINTS
- OPTIMIZER_IGNORE_PARALLEL_HINTS
- PARALLEL_DEGREE_POLICY
- PLSCOPE_SETTINGS
- PLSQL_CCFLAGS
- PLSQL_DEBUG
- PLSQL_OPTIMIZE_LEVEL
- PLSQL_WARNINGS
- RECYCLEBIN
- STATISTICS_LEVEL (Session only by using ALTER SESSION)
- TIME_ZONE (Session only by using 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.
Oracle Database Features That Are Not Supported

Autonomous Data Warehouse supports most of the features present in Oracle Database Enterprise Edition. To ensure the security and the performance of Autonomous Data Warehouse, some features are not supported.

The following Oracle Database features, options and packs are not supported in Autonomous Data Warehouse databases.

- Application containers
- Application contexts
- Clusters (groups of tables)
- Common users
- Dictionary-managed tablespaces
- Index-organized tables
- Logical standby databases
- Manual segment space management
- Manual undo management
- Oracle Cloud Management Pack for Oracle Database
- Oracle Data Masking and Subsetting Pack
- Oracle Database Lifecycle Management Pack for Oracle Database
- Oracle LogMiner
- Oracle On-Line Analytical Processing (OLAP) (Use analytic views instead. See Overview of Analytic Views for more information.)
- Oracle R Enterprise
- Oracle Real Application Testing
- Oracle Sharding
- Oracle Workspace Manager
- Root container (CDB$ROOT) access
- Transportable tablespaces
- Uniform extent allocation

Oracle Database Features with Limited Support

Autonomous Data Warehouse supports most of the features present in Oracle Database Enterprise Edition. To ensure the security and the performance of Autonomous Data Warehouse, some features have only limited support.

The following Oracle Database features, options and packs have limited support in Autonomous Data Warehouse databases.
### Limitations on the Use of Oracle Spatial and Graph

The following describes support for Oracle Spatial and Graph features in Autonomous Data Warehouse dedicated databases. To ensure the security and the performance of Autonomous Data Warehouse, some Oracle Spatial and Graph features are restricted.

⚠️ **Note:**

Oracle Autonomous Database does not include Oracle Spatial and Graph 3-Dimensional geometry types and related operators, functions, or utilities.

<table>
<thead>
<tr>
<th>Spatial and Graph Feature</th>
<th>Supported in Autonomous Database</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D Geometry data types</td>
<td>Yes</td>
<td>SDO_GEOMETRY Object Type</td>
</tr>
<tr>
<td>2D Spatial operators</td>
<td>Yes</td>
<td>Spatial Operators</td>
</tr>
<tr>
<td>2D Spatial functions</td>
<td>Yes</td>
<td>SDO_GEOM Package (Geometry)</td>
</tr>
<tr>
<td>Coordinate Transformation</td>
<td>Yes</td>
<td>SDO_CS Package (Coordinate System Transformation)</td>
</tr>
<tr>
<td>Spatial Utility Functions (except GML and KML conversion functions)</td>
<td>Yes</td>
<td>SDO_UTIL Package (Utility)</td>
</tr>
<tr>
<td>Spatial Aggregate Functions</td>
<td>Yes</td>
<td>Spatial Aggregate Functions</td>
</tr>
<tr>
<td>Spatial indexing and querying features</td>
<td>Yes</td>
<td>SQL Statements for Indexing Spatial Data</td>
</tr>
<tr>
<td>2D Linear Referencing System</td>
<td>Yes</td>
<td>SDO_LRS Package (Linear Referencing System)</td>
</tr>
<tr>
<td>Spatial and Graph Feature</td>
<td>Supported in Autonomous Database</td>
<td>More Information</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>----------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GeoJSON and JSON support for Spatial data types</td>
<td>Yes</td>
<td>JSON and GeoJSON Support in Oracle Spatial and Graph SDO_UTIL.TO_GEOJSON SDO_UTIL.TO_JSON SDO_UTIL.TO_JSON_VARCHAR</td>
</tr>
<tr>
<td>Location Tracking Server</td>
<td>Yes</td>
<td>SDO_TRKR Package (Location Tracking)</td>
</tr>
<tr>
<td>Spatial Analysis and Mining features</td>
<td>Yes</td>
<td>Spatial Analysis and Mining SDO_SAM Package (Spatial Analysis and Mining)</td>
</tr>
<tr>
<td>Flat Model for Point Cloud data</td>
<td>Yes</td>
<td>DO_PC_PKG.CLIP_PC_FLAT</td>
</tr>
<tr>
<td>RDF Graph</td>
<td>No</td>
<td>RDF Semantic Graph Overview</td>
</tr>
<tr>
<td>Property Graph</td>
<td>No</td>
<td>Spatial and Graph Property Graph Support Overview</td>
</tr>
<tr>
<td>Network Data Model</td>
<td>No</td>
<td>Network Data Model</td>
</tr>
<tr>
<td>GeoRaster</td>
<td>No</td>
<td>SDO_GEOR Package Reference</td>
</tr>
<tr>
<td>Routing Engine</td>
<td>No</td>
<td>Routing Engine</td>
</tr>
<tr>
<td>Geocoder</td>
<td>No</td>
<td>Geocoding Address Data</td>
</tr>
<tr>
<td>Spatial Visualization</td>
<td>No</td>
<td>Introduction to the Map Visualization Component</td>
</tr>
<tr>
<td>Topology Data Model</td>
<td>No</td>
<td>Topology Data Model Overview</td>
</tr>
<tr>
<td>Open Geospatial Consortium Web Services (WMS, WFS-T, C-SW, WCS, OpenLS)</td>
<td>No</td>
<td>SDO_WFS_PROCESS Package (WFS Processing) SDO_OLS Package (OpenLS)</td>
</tr>
<tr>
<td>GML</td>
<td>No</td>
<td>SDO_UTIL.TO_GMLGEO METRY</td>
</tr>
<tr>
<td>Point Cloud Object Type</td>
<td>No</td>
<td>SDO_PC_PKG Package (Point Clouds)</td>
</tr>
<tr>
<td>Triangulated Irregular Network (TIN) types</td>
<td>No</td>
<td>SDO_TIN_PKG Package (TINs)</td>
</tr>
</tbody>
</table>
Oracle Spatial and Graph relies on and uses Java in Oracle Database. Therefore, database sessions that use Oracle Spatial and Graph will experience reduced database-side performance when service maintenance operations are being performed on the database.

For details on Oracle Spatial and Graph, see Oracle Database 19c Spatial and Graph.

Limitations on the Use of Oracle Text

The following describes support for Oracle Text features in Autonomous Data Warehouse dedicated databases. To ensure the security and the performance of Autonomous Data Warehouse, some Oracle Text features are restricted.

<table>
<thead>
<tr>
<th>Oracle Text Feature</th>
<th>Supported in Autonomous Database</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>All logging, and APIs which perform logging such as ctx_report.query_log_summary</td>
<td>Not Supported</td>
<td>QUERY_LOG_SUMMARY</td>
</tr>
<tr>
<td>File and URL datastore</td>
<td>Not Supported</td>
<td>Datastore Type</td>
</tr>
<tr>
<td>CREATE INDEX with BIG_IO option</td>
<td>Not supported by default. ¹</td>
<td>Improved Response Time Using the BIG_IO Option of CONTEXT Index</td>
</tr>
<tr>
<td>OPTIMIZE_INDEX in rebuild mode</td>
<td>Not supported by default.(see Footnote 1)</td>
<td>OPTIMIZE_INDEX</td>
</tr>
</tbody>
</table>

¹ This is supported if you grant the privilege to create a trigger to the user (GRANT CREATE TRIGGER). You must also disable parallel DML at the session level (ALTER SESSION DISABLE PARALLEL DML).

For details on Oracle Text, see Oracle Text Application Developer's Guide.

Limitations on the Use of Oracle XML DB

The following describes support for Oracle XML DB features in Autonomous Data Warehouse dedicated databases. To ensure the security and the performance of Autonomous Data Warehouse, some Oracle XML DB are restricted.

If you migrate tables containing XMLType columns to Autonomous Data Warehouse using Oracle Data Pump, you need to convert to Non-Schema Binary XML prior to using Oracle Data Pump Export (expdp).

The following features are supported:

- 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

The following features are not supported or are only partially supported.

<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>
<tr>
<td>Index</td>
<td>XQuery Full Text Index</td>
<td>Yes</td>
<td>Indexing XML Data for Full-Text Queries</td>
</tr>
<tr>
<td>Index</td>
<td>Unstructured XMLIndex</td>
<td>No</td>
<td>XMLIndex Unstructured Component</td>
</tr>
<tr>
<td>Packages</td>
<td>XML DOM package</td>
<td>Yes</td>
<td>PL/SQL DOM API for XMLType (DBMS_XMLDOM)</td>
</tr>
<tr>
<td>Packages</td>
<td>XML Parser Package</td>
<td>Yes</td>
<td>PL/SQL Parser API for XMLType (DBMS_XMLPARSER)</td>
</tr>
<tr>
<td>Packages</td>
<td>XSL Processor</td>
<td>Yes</td>
<td>PL/SQL XSLT Processor for XMLType (DBMS_XSLPROCESSOR)</td>
</tr>
</tbody>
</table>

For details on Oracle XML DB, see Oracle XML DB Developer's Guide.

Limitations on the Use of SQL Commands

Autonomous Data Warehouse allows most of the SQL commands available in Oracle Database. To ensure the security and the performance of Autonomous Data Warehouse, some SQL commands are restricted.

This section provides a list of SQL command limitations that are required to protect security and for performance integrity in Autonomous Data Warehouse. Most of the standard SQL and PL/SQL syntax and constructs used with Oracle Database work in Autonomous Data Warehouse.
Note:

If you try to use a restricted SQL command the system generates this error:

ORA-01031: insufficient privileges

This error indicates that you are not allowed to run the SQL command in Autonomous Data Warehouse.

The following SQL commands and PL/SQL packages are not supported in Autonomous Data Warehouse databases:

- ADMINISTER KEY MANAGEMENT
- ALTER TABLESPACE
- DBMS_PIPE

SQL Commands with Restrictions in Autonomous Data Warehouse

The following DDL commands are available in Autonomous Data Warehouse with restrictions:

<table>
<thead>
<tr>
<th>SQL Command</th>
<th>Restrictions</th>
</tr>
</thead>
</table>
| ALTER PLUGGABLE DATABASE and ALTER DATABASE | Only the following clauses are allowed:  
DATAFILE AUTOEXTEND ON  
DATAFILE AUTOEXTEND OFF  
DATAFILE RESIZE  
DEFAULT EDITION  
SET TIME_ZONE |
<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 DEFAULT_COLLATION</td>
</tr>
<tr>
<td></td>
<td>SET EDITION</td>
</tr>
<tr>
<td></td>
<td>SET ISOLATION_LEVEL</td>
</tr>
<tr>
<td></td>
<td>SET ROW ARCHIVAL VISIBILITY</td>
</tr>
<tr>
<td></td>
<td>SET init-param = value</td>
</tr>
<tr>
<td></td>
<td>For a list of initialization parameters you can set in Oracle Autonomous</td>
</tr>
<tr>
<td></td>
<td>Database, see Modifying Database Initialization Parameters. Note that you</td>
</tr>
<tr>
<td></td>
<td>cannot set some of these parameters at the session level. To find out</td>
</tr>
<tr>
<td></td>
<td>whether a given parameter can be altered using ALTER SESSION SET, query the</td>
</tr>
<tr>
<td></td>
<td>ISSUES_MODIFIABLE column of the V$PARAMETER dynamic performance view.</td>
</tr>
<tr>
<td>ALTER SYSTEM</td>
<td>Only ALTER SYSTEM SET and ALTER SYSTEM KILL SESSION are allowed. SET can</td>
</tr>
<tr>
<td></td>
<td>only be used to set parameters listed in Modifying Database Initialization</td>
</tr>
<tr>
<td></td>
<td>Parameters.</td>
</tr>
<tr>
<td>ALTER TABLE</td>
<td>The following clauses are ignored:</td>
</tr>
<tr>
<td></td>
<td>allocate_extent</td>
</tr>
<tr>
<td></td>
<td>alter_iot</td>
</tr>
<tr>
<td></td>
<td>deallocate_unused</td>
</tr>
<tr>
<td></td>
<td>ilm</td>
</tr>
<tr>
<td></td>
<td>inmemory_table</td>
</tr>
<tr>
<td></td>
<td>logging</td>
</tr>
<tr>
<td></td>
<td>modify_LOB_storage</td>
</tr>
<tr>
<td></td>
<td>physical_attributes</td>
</tr>
<tr>
<td></td>
<td>shrink</td>
</tr>
<tr>
<td></td>
<td>For more information on ALTER TABLE, see Oracle Database SQL Language</td>
</tr>
<tr>
<td></td>
<td>Reference.</td>
</tr>
<tr>
<td>CREATE DATABASE LINK</td>
<td>In the USING 'connect string' clause, you must use the Easy Connect syntax</td>
</tr>
<tr>
<td></td>
<td>or provide a complete descriptor. You cannot use a network service name</td>
</tr>
<tr>
<td></td>
<td>because the tnsnames.ora file is not available for lookup.</td>
</tr>
<tr>
<td></td>
<td>Database links can only be used for TCP connections because TCPS connections</td>
</tr>
<tr>
<td></td>
<td>require a wallet.</td>
</tr>
</tbody>
</table>
### SQL Command Restrictions

<table>
<thead>
<tr>
<th>SQL Command</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE TABLE</td>
<td>XMLType tables are not allowed.</td>
</tr>
<tr>
<td></td>
<td>The following clauses are ignored:</td>
</tr>
<tr>
<td></td>
<td>cluster</td>
</tr>
<tr>
<td></td>
<td>ilm</td>
</tr>
<tr>
<td></td>
<td>inmemory_table</td>
</tr>
<tr>
<td></td>
<td>LOB_storage</td>
</tr>
<tr>
<td></td>
<td>logging</td>
</tr>
<tr>
<td></td>
<td>organization index</td>
</tr>
<tr>
<td></td>
<td>physical_properties</td>
</tr>
<tr>
<td></td>
<td>Additionally, use of the organization external clause has certain limitations, as described in Limitations on the Use of Manually Created External Tables. For more information on CREATE TABLE, see Oracle Database SQL Language Reference.</td>
</tr>
<tr>
<td>CREATE TABLESPACE</td>
<td>The SMALLFILE clause is not allowed.</td>
</tr>
</tbody>
</table>

### Limitations on the Use of Certain Data Types

Autonomous Data Warehouse allows most of the data types available in Oracle Database. To ensure the security and the performance of Autonomous Data Warehouse, some data types are restricted.

The following Oracle Database data types are not supported or are only partially supported in Autonomous Data Warehouse databases:

- The ROWID data type is not enabled by default, but you can enable it by setting the `ALLOW_ROWID_COLUMN_TYPE` initialization parameter to true.
- Large object (LOB) data types: only SecureFiles LOB storage is supported.
- Multimedia data types are not supported. (Oracle Multimedia is desupported in Oracle Database.)

For a list of Oracle data types see Oracle Database SQL Language Reference.

### Limitations on the Use of Manually Created External Tables

Instead of using the DBMS_CLOUD.CREATE_EXTERNAL_TABLE procedure to create an external table, you can use the SQL `CREATE TABLE...ORGANIZATION EXTERNAL` statement to manually create one. When you do so, certain limitations apply.

- The `ORACLE_LOADER` and `ORACLE_BIGDATA` access drivers are supported. The `ORACLE_DATAPUMP`, `ORACLE_HDFS` and `ORACLE_HIVE` access drivers are not supported.
- If you are specifying files stored in Cloud Object Storage as the data source location, you must use one of the URI formats described in Cloud Object Storage URI Formats.
• If you are specifying files stored in Cloud Object Storage as the data source location, you must include the **NOLOGFILE**, **NOSADFILE** and **NODISCARDFILE** clauses in the access parameters.
Oracle Autonomous Database includes the `DBMS_CLOUD` PL/SQL package to extend the functionality of Oracle Database to support autonomous and cloud operations.

**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 `atpc_user`:

```
GRANT EXECUTE ON DBMS_CLOUD TO atpc_user;
```

**Topics**

- `COPY_DATA` Procedure
- `CREATE_EXTERNAL_TABLE` Procedure
- `DELETE_ALL_OPERATIONS` Procedure
- `DELETE_FILE` Procedure
- `DELETE_OBJECT` Procedure
- `GET_OBJECT` Procedure
- `LIST_FILES` Function
- `LIST_OBJECTS` Function
- `PUT_OBJECT` Procedure
- `VALIDATE_EXTERNAL_TABLE` Procedure
- Format Parameter

**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,  

```
DBMS_CLOUD.COPY_DATA (table_name IN VARCHAR2, credential_name IN VARCHAR2 DEFAULT NULL, file_uri_list IN CLOB DEFAULT NULL, schema_name IN VARCHAR2 DEFAULT NULL, field_list IN CLOB DEFAULT NULL, format IN CLOB DEFAULT NULL, operation_id OUT NOCOPY NUMBER);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>The name of the target table on the database. The target table needs to be created before you run COPY_DATA.</td>
</tr>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>file_uri_list</td>
<td>Comma-delimited list of source file URIs. You can use wildcards in the file names in your URIs. The character &quot;*&quot; can be used as the wildcard for multiple characters, the character &quot;?&quot; can be used as the wildcard for a single character. The format of the URIs depend on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats.</td>
</tr>
<tr>
<td>schema_name</td>
<td>The name of the schema where the target table resides. The default value is NULL meaning the target table is in the same schema as the user running the procedure.</td>
</tr>
<tr>
<td>field_list</td>
<td>Identifies the fields in the source files and their data types. The default value is NULL meaning the fields and their data types are determined by the target table definition. This argument's syntax is the same as the field_list clause in regular Oracle external tables. For more information about field_list see Oracle® Database Utilities. For an example using field_list, see CREATE_EXTERNAL_TABLE Procedure.</td>
</tr>
<tr>
<td>format</td>
<td>The options describing the format of the source files. For the list of the options and how to specify the values see Format Parameter.</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>

CREATE_EXTERNAL_TABLE Procedure

This procedure creates an external table on files in the Cloud. This allows you to run queries on external data from Autonomous Database.

Syntax

DBMS_CLOUD.CREATE_EXTERNAL_TABLE (table_name IN VARCHAR2, credential_name IN VARCHAR2, file_uri_list IN CLOB,
CREATE_EXTERNAL_TABLE Procedure

```
column_list IN CLOB,
field_list IN CLOB DEFAULT,
format IN CLOB DEFAULT);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>The name of the external table.</td>
</tr>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>file_uri_list</td>
<td>Comma-delimited list of source file URIs. You can use wildcards in the file names in your URIs. The character &quot;*&quot; can be used as the wildcard for multiple characters, the character &quot;?&quot; can be used as the wildcard for a single character. The format of the URIs depend on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats.</td>
</tr>
<tr>
<td>column_list</td>
<td>Comma-delimited list of column names and data types for the external table.</td>
</tr>
<tr>
<td>field_list</td>
<td>Identifies the fields in the source files and their data types. The default value is NULL meaning the fields and their data types are determined by the column_list parameter. This argument's syntax is the same as the field_list clause in regular Oracle external tables. For more information about field_list see Oracle® Database Utilities.</td>
</tr>
<tr>
<td>format</td>
<td>The options describing the format of the source files. For the list of the options and how to specify the values see Format Parameter.</td>
</tr>
</tbody>
</table>

Usage Notes

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.

Example

```
BEGIN
  DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
    table_name => 'WEATHER_REPORT_DOUBLE_DATE',
    credential_name => 'OBJ_STORE_CRED',
    file_uri_list => '&base_URL/Charlotte_NC_Weather_History_Double_Dates.csv',
    format => json_object('type' value 'csv', 'skipheaders' value '1'),
    field_list => 'REPORT_DATE DATE''mm/dd/yy'',
                 'REPORT_DATE_COPY DATE ''yyyy-mm-dd'',
                 'ACTUAL_MEAN_TEMP',
                 'ACTUAL_MIN_TEMP',
                 'ACTUAL_MAX_TEMP',
                 'AVERAGE_MIN_TEMP',
                 'AVERAGE_MAX_TEMP',
                 'AVERAGE_PRECIPITATION',
    column_list => 'REPORT_DATE DATE,
                   REPORT_DATE_COPY DATE,
```

```
ACTUAL_MEAN_TEMP NUMBER,
ACTUAL_MIN_TEMP NUMBER,
ACTUAL_MAX_TEMP NUMBER,
AVERAGE_MIN_TEMP NUMBER,
AVERAGE_MAX_TEMP NUMBER,
AVERAGE_PRECIPITATION NUMBER');
END;
/

SELECT * FROM WEATHER_REPORT_DOUBLE_DATE where
  actual_mean_temp > 69 and actual_mean_temp < 74

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 atpc_user:

```
GRANT WRITE ON DIRECTORY data_pump_dir TO atpc_user;
```

DELETE_OBJECT Procedure

This procedure deletes the specified object on object store.

Syntax

```
DBMS_CLOUD.DELETE_OBJECT (  
  credential_name      IN VARCHAR2,  
  object_uri           IN VARCHAR2);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>object_uri</td>
<td>Object or file URI for the object to delete. The format of the URI depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats.</td>
</tr>
</tbody>
</table>

GET_OBJECT Procedure

This procedure reads an object from Cloud Object Storage and copies it to Autonomous Database. The maximum file size allowed in this procedure is 5 gigabytes (GB).

Syntax

```
DBMS_CLOUD.GET_OBJECT (  
  credential_name      IN VARCHAR2,  
  object_uri           IN VARCHAR2,  
  directory_name       IN VARCHAR2,  
)
```
file_name       IN VARCHAR2 DEFAULT NULL,
startoffset     IN NUMBER DEFAULT 0,
endoffset       IN NUMBER DEFAULT 0,
compression     IN VARCHAR2 DEFAULT NULL);

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>object_uri</td>
<td>Object or file URI. The format of the URI depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage 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 atpc_user:

`GRANT WRITE ON DIRECTORY data_pump_dir TO atpc_user;`

Example

```
BEGIN
    DBMS_CLOUD.GET_OBJECT(
       credential_name => 'OBJ_STORE_CRED',
       object_uri => 'https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/idthydc0kinr/mybucket/cwallet.sso',
       directory_name => 'DATA_PUMP_DIR');
END;
/
```

In this example, object_uri is an Oracle Cloud Infrastructure Swift URI that specifies the cwallet.sso file in the mybucket bucket in the us-phoenix-1 region. (idthydc0kinr is the object storage namespace in which the bucket resides.) For information about the supported URI formats, see Cloud Object Storage URI Formats.
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**

```sql
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 `atpc_user`:

  ```sql
  GRANT READ ON DIRECTORY data_pump_dir TO atpc_user;
  ```

**Example**

This function returns a row for each file. For example, use the following query to use this function:

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

<table>
<thead>
<tr>
<th>OBJECT_NAME</th>
<th>BYTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>cwallet.sso</td>
<td>2965</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**

```sql
DBMS_CLOUD.LIST_OBJECTS (  
    credential_name      IN VARCHAR2,  
    location_uri         IN VARCHAR2  
)  
RETURN TABLE;
```
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>location_uri</td>
<td>Object or file URI. The format of the URI depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats.</td>
</tr>
</tbody>
</table>

Example

This function 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://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/idthydc0kinr/mybucket/');
```

<table>
<thead>
<tr>
<th>OBJECT_NAME</th>
<th>BYTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>cwallet.sso</td>
<td>2965</td>
</tr>
</tbody>
</table>

PUT_OBJECT Procedure

This procedure copies a file from Autonomous Database to the Cloud Object Storage. The maximum file size allowed in this procedure is 5 gigabytes (GB).

Syntax

```sql
DBMS_CLOUD.PUT_OBJECT ( 
    credential_name      IN VARCHAR2, 
    object_uri           IN VARCHAR2, 
    directory_name       IN VARCHAR2, 
    file_name            IN VARCHAR2);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential_name</td>
<td>The name of the credential to access the Cloud Object Storage.</td>
</tr>
<tr>
<td>object_uri</td>
<td>Object or file URI. The format of the URI depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats.</td>
</tr>
<tr>
<td>directory_name</td>
<td>The name of the directory on the Autonomous Data Warehouse dedicated 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 `atpc_user`:

```
GRANT READ ON DIRECTORY data_pump_dir TO atpc_user;
```

Usage Note

Oracle Cloud Infrastructure object store does not allow writing files into a public bucket without supplying credentials (Oracle Cloud Infrastructure allows users to download objects from public buckets). Thus, you must supply a credential name with valid credentials to store an object in an Oracle Cloud Infrastructure public bucket using `PUT_OBJECT`.

VALIDATE_EXTERNAL_TABLE Procedure

This procedure validates the source files for an external table, generates log information, and stores the rows that do not match the format options specified for the external table in a `badfile` table on Autonomous Database. The overloaded form enables you to use the `operation_id` parameter.

Syntax

```
DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE (  
    table_name      IN VARCHAR2,  
    schema_name     IN VARCHAR2 DEFAULT,  
    rowcount        IN NUMBER DEFAULT,  
    stop_on_error   IN BOOLEAN DEFAULT);
```

```
DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE (  
    table_name      IN VARCHAR2,  
    operation_id    OUT NOCOPY NUMBER,  
    schema_name     IN VARCHAR2 DEFAULT NULL,  
    rowcount        IN NUMBER DEFAULT 0,  
    stop_on_error   IN BOOLEAN DEFAULT TRUE);
```

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>table_name</code></td>
<td>The name of the external table.</td>
</tr>
<tr>
<td><code>operation_id</code></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><code>schema_name</code></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>
</tbody>
</table>
### Format Parameter

The format parameter is used in several DBMS_CLOUD subprograms to specify the format of source files.

The two ways to specify the format parameter are:

- `format => '{"format_option" : "format_value" }'`

- `format => json_object('format_option' value 'format_value')`

**Examples:**

- `format => json_object('type' VALUE 'CSV')`

To specify multiple format options, separate the values with a ",".

**For example:**

- `format => json_object('ignoremissingcolumns' value 'true', 'removequotes' value 'true', 'dateformat' value 'YYYY-MM-DD-HH24-MI-SS', 'blankasnull' value 'true')`

<table>
<thead>
<tr>
<th>Format Option</th>
<th>Description</th>
<th>Syntax</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>blankasnull</td>
<td>When set to true, loads fields consisting of spaces as null.</td>
<td>blankasnull : true</td>
<td>False</td>
</tr>
<tr>
<td>charsetset</td>
<td>Specifies the character set of source files</td>
<td>charsetset: string</td>
<td>Database character set</td>
</tr>
<tr>
<td>Format Option</td>
<td>Description</td>
<td>Syntax</td>
<td>Default</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>compression</td>
<td>Specifies the compression type of the source file.</td>
<td>compression: auto</td>
<td>gzip</td>
</tr>
<tr>
<td>conversionerrors</td>
<td>If a row is rejected because of data type conversion errors, the related columns are stored as null or the row is rejected.</td>
<td>conversionerrors: reject_record</td>
<td>store_null</td>
</tr>
<tr>
<td>Format Option</td>
<td>Description</td>
<td>Syntax</td>
<td>Default</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>dateformat</td>
<td>Specifies the date format in the source file. The format option AUTO searches for the following formats:</td>
<td>dateformat: string</td>
<td>Database date format</td>
</tr>
<tr>
<td></td>
<td>J</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM-DD-YYYYBC</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM-DD-YYYY</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYY-MMDD HHMISS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYY-MMDD HHMISS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYY.DD</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYY-MM-DD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>delimiter</td>
<td>Specifies the field delimiter</td>
<td>delimiter: character</td>
<td></td>
</tr>
<tr>
<td>escape</td>
<td>The character &quot;&quot; is used as the escape character when specified.</td>
<td>escape: true</td>
<td>False</td>
</tr>
<tr>
<td>ignoreblanklines</td>
<td>Blank lines are ignored when set to true.</td>
<td>ignoreblanklines:</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>ignoremissingcolumns</td>
<td>If there are more columns in the field_list than there are in the source files, the extra columns are stored as null.</td>
<td>ignoremissingcolumns:</td>
<td>False</td>
</tr>
<tr>
<td></td>
<td></td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>language</td>
<td>Specifies a language name (for example, FRENCH), from which locale-sensitive information can be derived.</td>
<td>language: string</td>
<td>Null</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See Locale Data in Oracle Database Globalization Support Guide for a listing of Oracle-supported languages.</td>
<td></td>
</tr>
<tr>
<td>numericcharacters</td>
<td>Specifies the characters to use as the group separator and decimal character.</td>
<td>numericcharacters:</td>
<td>&quot;,,&quot;</td>
</tr>
<tr>
<td></td>
<td>decimal_character: The decimal separates the integer portion of a number from the decimal portion.</td>
<td>'decimal_character group_separator'</td>
<td>See NLS_NUMERIC_CHARACTERS in Oracle Database Globalization Support Guide for more information.</td>
</tr>
<tr>
<td></td>
<td>group_separator: The group separator separates integer groups (that is, thousands, millions, billions, and so on).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>numberformat</td>
<td>Specifies the number format model. Number format models cause the number to be rounded to the specified number of significant digits. A number format model is composed of one or more number format elements. This is used in combination with numericcharacters.</td>
<td>numberformat:</td>
<td>Default is derived from the setting of the NLS_TERRITORY parameter</td>
</tr>
<tr>
<td></td>
<td>number_format_model</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>See Number Format Models in SQL Language Reference for more information.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format Option</td>
<td>Description</td>
<td>Syntax</td>
<td>Default</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------------</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>
<td>Null meaning no quote</td>
</tr>
<tr>
<td>recorddelimiter</td>
<td>Specifies the record delimiter.</td>
<td>recorddelimiter: character</td>
<td>newline</td>
</tr>
<tr>
<td>rejectlimit</td>
<td>The operation will error out after specified number of rows are rejected.</td>
<td>rejectlimit: number</td>
<td>0</td>
</tr>
<tr>
<td>removequotes</td>
<td>Removes any quotes that are around any field in the source file.</td>
<td>removequotes: true</td>
<td>False</td>
</tr>
<tr>
<td>skipheaders</td>
<td>Specifies how many rows should be skipped from the start of the file.</td>
<td>skipheaders: number</td>
<td>0 if not specified, 1 if specified without a value</td>
</tr>
<tr>
<td>territory</td>
<td>Specifies a territory name to further determine input data characteristics.</td>
<td>territory: string</td>
<td>Null</td>
</tr>
</tbody>
</table>

By default, DBMS_CLOUD tries to automatically find the correct newline character as the delimiter. It first searches the file for the Windows newline character "\r\n". If it finds the Windows newline character, this is used as the record delimiter for the file. If a Windows newline character is not found, it searches for the UNIX/Linux newline character "\n" and if it finds one it uses it as the record delimiter for the file. Specify this parameter explicitly if you want to override the default behavior, for example:

```json
format =>
json_object('recorddelimiter' VALUE '"\r\n")
```
<table>
<thead>
<tr>
<th>Format Option</th>
<th>Description</th>
<th>Syntax</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestampformat</td>
<td>Specifies the timestamp format in the source file. The format option AUTO searches for the following formats:</td>
<td>timestampformat: <code>string</code></td>
<td>Database timestamp format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYY-MM-DD HH:MI:SS.FF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYY-MM-DD HH:MI:SS.FF3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM/DD/YYYY HH:MI:SS.FF</td>
<td></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: <code>string</code></td>
<td>Database timestamp with local timezone format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DD Mon YYYY HH:MI:SS.FF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TZR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM/DD/YYYY HH:MI:SS.FF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TZR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYY-MM-DD HH:MI:SS+/-TZR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYY-MM-DD HH:MI:SS.FF3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DD.MM.YYYY HH:MI:SS TZR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>timestamptzformat</td>
<td>Specifies the timestamp with timezone format in the source file. The format option AUTO searches for the following formats:</td>
<td>timestamptzformat: <code>string</code></td>
<td>Database timestamp with timezone format</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DD Mon YYYY HH:MI:SS.FF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TZR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM/DD/YYYY HH:MI:SS.FF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TZR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYY-MM-DD HH:MI:SS+/-TZR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>YYYY-MM-DD HH:MI:SS.FF3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DD.MM.YYYY HH:MI:SS TZR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>trimspaces</td>
<td>Specifies how the leading and trailing spaces of the fields are trimmed. See the description of <code>trim_spec</code> in Oracle Database Utilities.</td>
<td>trimspaces: <code>rtrim</code></td>
<td>notrim</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ltrim</td>
<td>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: <code>true</code></td>
<td>False</td>
</tr>
<tr>
<td>Format Option</td>
<td>Description</td>
<td>Syntax</td>
<td>Default</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>type</td>
<td>Specifies the source file type. Cannot be specified together with delimiter or quote. See the description of CSV in Oracle Database Utilities.</td>
<td>type: csv</td>
<td>csv with embedded</td>
</tr>
</tbody>
</table>